UTRECHT UNIVERSITY

MASTER THESIS

Integrating Agent Planning and Emotions

Author:
Kevin KESSELS

First Supervisor:
Dr. Frank DIGNUM

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Artificial Intelligence
Department of Information and Computing Sciences

ICA-5575850

July 17, 2017

UTRECHT UNIVERSITY

Abstract

Science
Department of Information and Computing Sciences

Master of Science

Integrating Agent Planning and Emotions

by Kevin KESSELS

Agent planning and emotion simulation are two aspects of agent based research that have largely been kept separate. This is despite plenty of literature showing the significant impact that emotions have on human decision making. In this thesis we determine the feasibility and benefits of generic integration between basic emotion systems and planners. We believe that by enabling interaction between emotions and automated planners we can more accurately mimic the human decision making process, allowing for more realistic solutions to a potentially greater set of problems.

From our research we have derived a model to facilitate the integration of existing emotion and planning systems. The model specifies how, several basic planning components such as actions and goals, can be used to allow emotions to influence, and be influenced by, a planner. Lastly, we implemented the model as a prototype which enhances a planner with emotions from an emotion system. Our results show that a basic scenario with emotions such as anger and fear can already enhance the planning significantly. It also enabled the planner to resolve a social scenario using purely emotional reasoning. Lastly, we discuss the cases in which we believe the model to be applicable.

Contents

Chapter 1

Introduction

1.1 Motivation

Computer science and artificial intelligence are fast growing and changing domains that have a wide variety of applications. Automated planning, which involves a planning system's ability to generate a sequence of actions that will accomplish a given task or goal, is a research area that has a large number of applications ranging from virtual agents in video games to unmanned vehicles sent out to explore the solar system. In general this form of planning uses reasoning in a purely logical fashion. Information is gathered from the environment, whether virtual or physical, and then processed by applying decision making steps that result in an executable plan. By logical we mean that these decision making steps are based on clear formal rules.

However, there are other factors that contribute to the decision making process. This can be observed in humans in the form of emotions. In this case emotions are not limited to a guidance role, and can even take complete control. It has come to the attention of the planning community that emotions play a major role in our own decision making process, making it an interesting topic to explore.

At the same time there is already a vast amount of knowledge regarding emotions from other fields of research. This deep understanding of emotions has lead to a wide variety of models and representations that aim to identify the roles and effects that emotions play in the human psyche. For example, emotions can heavily bias our decision making process, leading us to choices that would not even be considered under different emotional circumstances. The detailed analysis that has gone into these emotional models has lead to evidence going as far as to suggest that emotions might be the dominant factor in our decision making process.

Currently available planners come in many varieties, however, what they lack is integration of logical and emotional reasoning. Most planners are purely logical, and therefore are unable to generate an acceptable plan for even simple social scenarios, were emotions are often integral to the decisions a human makes. Especially in the case of agents that interact with humans, where it is almost always obvious to the human that they are interacting with an agent. Without emotions the reactions and behaviors of an agent are purely calculated and logical, which is seen as unrealistic human behavior. When, for example, two agents dislike each other but they have to get along to solve a problem, a logical planner would find a solution where they work together. However, their dislike can realistically lead to difficulty or maybe even the inability to solve the problem together. A human

observing such a scenario will find a solution in which the agents simply work together unsatisfactory since this cannot be applied to the real world. There are planners available that take emotions into account, however, they are limited to specific aspects such as reactionary emotions. This means that if our agent performs an action that negatively affects another then the negatively affected agent becomes angry at our agent and will refuse to help him. This kind of emotional reasoning, however, is still separated from the actual planning logic, and follow up planning steps with the same agents will not take these emotions into account. Especially in social scenarios where agents have history or affiliations this compartmentalized reasoning prevents planners from being able to take into account all the relevant factors, leading to unrealistic interactions. Simply allowing agents to have access to a wide variety of behaviors such as aggressive and avoidance behaviors does not resolve the issues. These behaviors have to be handled correctly, as applying these behaviors at inappropriate times will also result in odd and unrealistic behavior.

The prospect of how emotions can be integrated with automated planning systems has become more interesting now that the automation field has matured enough to make such an addition a worthwhile endeavor. Logical reasoning has established itself as a first-class citizen of the automated planning world. However, it is becoming evident that emotional reasoning is just as important. Especially in the agent planning domain there is a large set of planning problems that are currently difficult or impossible to solve. When social dynamics are a key component, current automated planners do not properly address the planning-intricacies at play. This creates a very interesting opportunity for exploring the benefits of modeling emotions in such a manner that they can work together with existing automated planning systems, enriching them with the knowledge of emotional reasoning.

1.2 Research Questions

Automated planners have difficulties in solving complex social scenarios in an acceptable and realistic manner. A key component of social scenarios is the fact that humans, in addition to logical reasoning, have emotional responses [14]. However, automated planners are generally limited to logical reasoning, rarely making use of even basic emotional reactions. We believe that adding emotional reasoning to automated planners could help them to overcome the difficulties that they currently face when planning in social scenarios. However, there are already a great number of planners and emotion simulation models. Therefore, rather than attempting to create a new planner that incorporates emotions from the start, we seek to integrate emotions with existing automated planners. This allows us to benefit from the research that has been done in both the planning and emotion fields. The resulting integration model can then be used with existing automated planners to enhance their ability to solve complex problems while also generating more realistic solutions.

Research Question 1 *Can the integration of emotional reasoning allow existing automated planners to realistically solve social scenarios?*

Research Question 2 *To what degree can the exploration of emotion models and planners lead to generic integration methods?*

1.3 Methodology

The goal of this thesis is to explore the possibilities for incorporating emotions with existing planning systems. First we will explore the the state-of-the-art in terms of planners and emotion representation and simulation. We follow this up with an analysis of emotions and how they influence human decision making. Next, we have to determine generic planner points of influence that we can use to simulate the emotional effects. We then attempt to define, by specifying a model, how each emotion can use the planning points of influence to implement a generic integration between planner and emotion system. Lastly, we create a prototype that we use, in combination with several scenarios, to determine the feasibility of our model's integration, while also assessing the planner's enhanced problem solving capabilities.

1.4 Thesis Outline

In Chapter 2 we will provide background information on the various available agent emotion and planning techniques and frameworks. Then, in Chapter 3 we will delve into the function and effects of emotions for humans. Next, in Chapter 4 we present a model that specifies how to integrate the emotion and planning aspects. Chapter 5 then follows up with an explanation of implementation details such as the integration planning algorithm and how the planner and emotions influence a shared (emotional) state. A prototype of the model is described in Chapter 6, together with several experiments that explore the benefits and feasibility of incorporating emotions into an existing planner. Lastly, we conclude with Chapter 7, which also includes a discussion of potential future work.

Chapter 2

Background Information and Related Work

2.1 Introduction

In this chapter we will first explore the emotion side of reasoning and planning. After a discussion of agent emotions we delve into some existing emotional models that attempt to simulate emotions in order to learn about the techniques utilized to turn agent observations into relevant emotions. Next we take a more detailed look at the Fatima framework, which provides a basis for emotion theory testing, making it a useful tool for exploring the benefits and downsides of the many emotional models that exist. Lastly, we follow up with a section on cognitive and emotional reasoning to determine the similarities between both types of decision making.

Then we switch to the planning domain. We start off by taking a look at knowledge reasoning, quickly moving on to planners. We classify automated planners according to their domain applicability. Domain specific, domain independent, and domain configurable planners each have their merits. With the aid of an example we seek to determine which type of planner is appropriate for testing the potential of combining planners with emotions. The chapter is concluded with a short summary that states which aspects will be further explored.

2.2 Agent Emotions

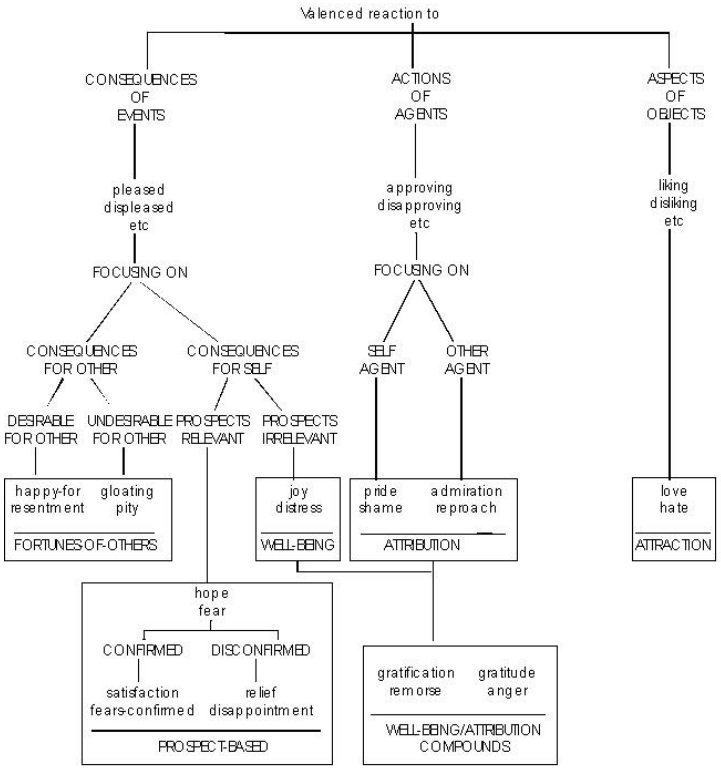
One of the interesting areas of agent related research is emotions. There are many models and techniques for simulating emotions. These can range from representing basic emotions to the complex social norms and values that govern our everyday lives. There are a large amount of inputs required to allow an agent to be able to make complex problem solving decisions. As more inputs are considered the complexity of the decision process scales even faster, meaning a delicate balance has to be maintained [34]. We take a look at several emotion theories that attempt to accurately simulate how emotions are generated and handled in order to gain insight into how emotions can be utilized to benefit agent planning.

2.2.1 Emotion Models

There are a large variety of emotion models. The OCC theory of emotions, proposed by Ortony et al. [29], is a model that considers emotions a reaction to the perception of the world. It bases the strength of an emotion on

this cognition and interpretation of the world around an agent, allowing it to observe events, actions, and objects and have an emotional response [4]. In addition, the model also enables the consideration of consequences for self and others via desirability. For example, if an agent observes someone harassing their friend they can become angry as this is seen as unfair treatment of their friend.

FIGURE 2.1: OCC Model Structure of Emotion Types [29]

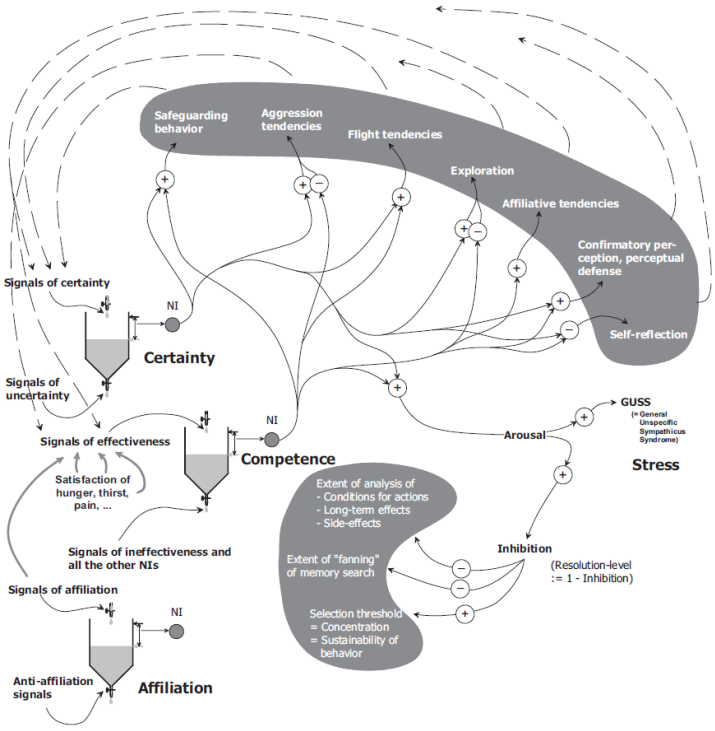


OCC distinguishes between 22 emotions divided up into 6 categories in a tree like structure. As can be seen in Figure 2.1 these categories are fortunes of others, well-being, attribution, attraction, prospect-based, and well-being/attribution-compounds. The categories are based on whether the reaction is based on an event, action, or object, although some combinations are possible. The emergence of an emotion is determined via the evaluation of three intensity variables: desirability, praiseworthiness, and appealingness. If these result in a positive value then the emergence of a relevant emotion is possible. This, in combination with a set of global and local intensity variables such as proximity and unexpectedness, then results in a function for the potential of that emotion emerging [4]. From this function an intensity function can also be obtained. This model closely follows how emotions are evoked in the real world, however, it is highly complex and often unfeasible to implement a domain independent algorithm that can derive the appropriate emotions from the actions, events, and world around an agent. Ortony, Clore, and Collins themselves provide no formalization for each emotion, instead only giving a few examples. Therefore, either the implementation has to make some simplifications, losing a portion of the interesting parts of the model, or a solution is needed to facilitate the practical creation of a complete implementation. Still, it is a good model

for representing and generating emotions and we will take a closer look at the emotion framework Fatima, which uses this model.

A model that emphasizes how emotions are results of basic human action regulation is the Psi theory. These motives are the need for energy, water, pain-avoidance, affiliation, certainty, and competence [1]. These needs are then translated to emotions based on an agent's ability to fulfill them. However, emotions are not their own entity in Psi. Instead they are a modulation of a cognitive-motivational process [8]. If, for example, an agent is prevented from reaching its goal, it can take on aggressive tendencies. In Psi emotions are a change in thinking, perceiving, and other cognitive processes about the environment and the agent's motives. The Psi theory benefits from being relatively straightforward in what generates the 'emotions'.

FIGURE 2.2: Psi model. "Emotions as modulation parameters and behavior tendencies. Note: For better clarity, the figure does not include all feedback loops, for example, those from the behavior tendencies back to the certainty and competence tanks. Certainly, successful exploration, for instance, is connected to a refilling of the certainty tank." [8]



In figure 2.2 we can see how the Psi model is structured. In the bottom left there are three tanks: certainty, competence, and affiliation. The deviation of a need from its set point determines the strength of the need indicators, which are linked to the upper and lower cloud. The upper cloud mainly contains behavior tendencies involving the outside world, while the lower cloud is focused on internal processes such as effects of actions and selection threshold. These motives can be adapted to the circumstances presented by the environment of the agent, allowing for accurate simulation of the basic emotions such as happiness, anger, and fear. While the motives are partially based on social factors such as the need for affiliation, it is focused

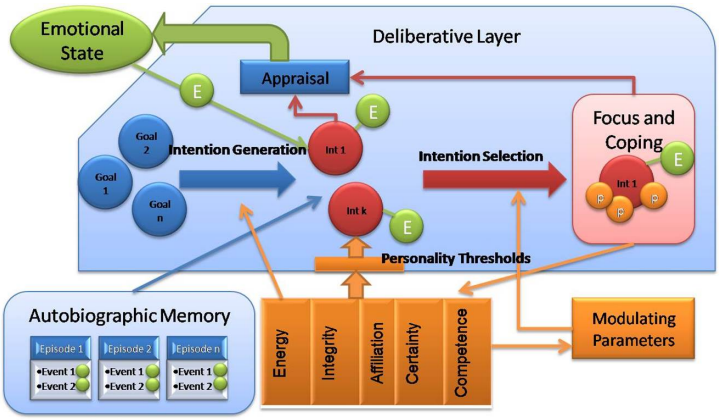
on generating emotions that relate to survival rather than the wide range of complex society forming social emotions and norms that humans have. Adding these emotions as modulations to Psi would be a major endeavor, and therefore is also a significant limitation when a wide range of emotions is to be explored and utilized. This means that not all emotion models are capable of simulating the kind of emotions that are often involved in social scenarios.

2.2.2 FAtiMA

There are also fully fledged frameworks that try to facilitate the implementation of emotion generation theories such as the ones mentioned previously. FatiMa is a generic and flexible framework which integrates the OCC theory of emotions in order to generate emotions. It has a core that contains the minimum functionality to implement and compare different appraisal theories [7]. This core can then be extended with additional components to facilitate desired functionality.

Figure 2.3 shows the reactive and deliberative appraisal, as well as the action selection layer. The reactive appraisal handles quick reactions to events based on predefined emotions. The deliberative appraisal is slower but also takes into account the likelihood of intention success or failure in order to generate an emotional response. The generated emotions are also stored in the autobiographic memory for future reference. The reactive and deliberative appraisals handle all of OCC's 22 emotions, and after the appraisal phase they perform practical reasoning. In this phase the reactive layer links emotions to actions, these are known as action tendencies and define impulse reactions. The deliberative layer generates goal driven behavior, making use of active pursuit goals and interest goals. Active-pursuit goals are followed to achieve a certain state while interest goals are maintained continuously to avoid threatening situations [19]. The derived emotions have a type, valence, and intensity which all decay over time. These basic aspects can also be extended with additional modulation factors. It additionally also handles each as a separate component, allowing for mixed emotions to be experienced.

FIGURE 2.3: FAtiMA Architecture [19]



Fatima also has a continuous planner with problem and emotion focused coping. Problem focused coping is the standard method of planning, where a set of actions that achieve a desired goal is created. Emotion focused coping on the other hand changes an agent's interpretation of the circumstances by, for example, lowering the importance of a goal [19].

Besides Fatima's combination of OCC-based cognitive appraisal and a planner utilizing problem and emotion focused coping it also provides the ability to test other appraisal theories. The core of Fatima can be extended with extra modules and the appraisal theory can be substituted for a different one. The benefits of testing a variety of emotional appraisal theories are great, however, Fatima does have some significant downsides. Firstly, Fatima does not justify the appearance of goals or desires, which means that it is difficult to know when these will appear, and consequently, influence the planning and emotions. This links to another difficulty which is the complexity of authoring an agent. An XML definition of goals, emotional reactions, actions and effects, and action tendencies all have to be specified [19]. This makes it very difficult to create desired behavior and this difficulty is compounded by the fact that there is little theoretical reasoning for defining these values, leading to a trial and error process for creating a desired agent. Lastly, Fatima handles emotions in a purely reactive manner, evaluating the current situation and reacting to events in order to simulate emotions. It does not incorporate emotions into a long-term plan, yet there emotions such as jealousy and indignation that can influence long-term planning significantly.

2.2.3 Cognitive and Emotional Reasoning

Emotions and cognitive decision-making have long been considered to be best kept separate. However, research into the interplay of these two seemingly conflicting forces has lead to some interesting discoveries. Learning or adopting a belief is based on more than cognitive judgment [14]. Emotions and feelings shape belief. They also help to direct our attention, motivate our interests, compel our assent, and alert us to risks. There are valid concerns for keeping emotions separate as they can undermine the intellectual quality and objectivity of decisions. At the same time there is also evidence that emotions are the dominant factor in the decision-making process, shaping decisions via goal activation. For example, the disgust emotion can create an implicit goal to expel the object that is the source of disgust and prevent the agent from taking in any further objects of that kind [17]. Studies on happiness and positive emotions have shown that there is a correlation between being in a positive mood and making better decisions [20]. Similarly, negative emotions tend to cause less efficient decision making due to risk-reduction taking priority over everything. This interplay between emotions and planning is important to explore as it has the potential to greatly improve the accuracy and realism of techniques for both fields.

One computational model that works with cognitive reasoning is known as the BDI model, which stands for beliefs, desires, and intentions [27]. This model allows for practical reasoning in terms of deliberation and means end reasoning. Because it makes use of beliefs, desires, and intentions it is easier to understand how the emotions are generated when compared to

other models. For example, if an agent believes there to be water in the fridge, has the desire to stay hydrated, and the intention to stay healthy, then the agent would act on its intentions and get some water from the fridge and drink it. However, it does not provide the ability to learn from experiences or adapt to the environment, limiting the scope of problems it can tackle. Emotions can allow such a model to, for example, learn from one's mistakes in the form of emotions such as embarrassment or shame. By incorporating emotions which are generated as a reaction to what is happening around the agent, a reasoning model such as this could potentially be greatly enhanced.

2.3 Knowledge Reasoning

For completeness sake we also want to mention an agent's reasoning abilities, however, we will not be making explicit use of this ability when determining emotion-planning integration methods. An agent's ability to reason about it's own knowledge is a valuable asset since it enables new knowledge to be generated from previously acquired knowledge. This ability to acquire more knowledge in an internal process can allow an agent to plan and simulate emotions more effectively. An agent can deduce, without having to directly observe why, that frightened people leaving a room means that that room should be avoided. While classical planning can handle a decent variety of problems, there are more advanced techniques that reason using the knowledge that an agent collects by observing the world around it in order to hypothesize about the outcomes of possible actions. This increased ability to reason about a situation facilitates the similarly intricate interactions that emotions can add to planning.

2.4 Planners

A planner is a system that can receive a planning problem as input and then return a plan to solve it. The input generally consists of a description of the system, the initial state, and a goal to be reached [24]. A plan, consisting of a sequence of actions to take, is created from this input. Automated planners can be classified according to how they handle the domain that they are made to function in. This classification consists of domain specific, domain independent, and domain configurable planners. We will explore each classification with some examples of a planner of that class, while also discussing the advantages and disadvantages of each class of planner.

2.4.1 Domain Specific Planners

Domain specific planners make use of domain specific aspects, such as a planner for an airline company not needing to follow the roads when generating routes. This knowledge allows for an efficient solver, and it is this type of planner that is almost always used in practical applications. However, it does mean that every planner of this class is uniquely designed for its domain and therefore has little use in other domains.

A good example of a complex domain specific planner is the Spike [16] planning and scheduling software developed for the Hubble Space Telescope (HST). Since there are 10,000 to 30,000 observations scheduled per year the purpose of Spike is to allow the telescope's time to be utilized as efficiently as possible. However, besides the large number of scheduling constraints there are also a large variety of planning constraints. The software has a planning and scheduling component, but we will focus on the planning side of things which is called the Transformation system. Examples of planning tasks are the matching of target visibility conditions, grouping observation to minimize overhead, and choosing specific implementation scenarios [16]. On top of this, even though the HST works in a two months in advance preplanned mode, there are disruptions such as an opportunity to observe a supernova, or a loss of communication with the ground. Such a disruption means the plan has to be modified to efficiently include the new operations. There are countless more factors that have to be taken into account when planning for the HST, and without having a planner specifically designed to handle these circumstances the planning would take too long or have to create a very rough plan with a large amount of inefficiently planned actions.

To tackle this planning problem the planner handles the plan construction as a constrained optimization problem. It makes use of suitability functions, which are described the the next paragraph, and Spike's temporal constraint mechanism to achieve path consistency [16]. The temporal constraint mechanism in turn uses a heuristic repair-based planning technique called multistart stochastic repair [16]. This techniques uses a three step process. First, it makes an initial guess of activities and their time slots based on heuristics derived from extensive experimentation with thousands of combinations of heuristics. This plan will generally have plenty of constraint violations and resource capacity overloads. Next, it applies a heuristic repair technique based on a neural network architecture that has been refined into a simple symbolic form over the years. The repair heuristic seeks to eliminate constraint violations until successful or an established effort threshold is met. Lastly, remaining conflicts are solved by relaxing constraints or removing activities with constraint violations until a feasible plan is found. Since the heuristics are stochastic the general strategy is to repeat the three steps as often as time permits and then select the best solution. This in conjunction with desirable "anytime" characteristics [36] means a feasible plan can be obtained at any time by removing any remaining constraint violating activities.

The heuristics in turn pay careful attention to the suitability function values to optimize the suitability of the final schedule. The suitability functions take the form of a constraint such as: "Schedule A_j as soon as possible after the end of A_i, but starting no sooner than x minutes afterwards and ending no later than $x + y$ minutes afterwards." [16].

As can be seen there is a vast amount of work that goes into creating a domain specific planner, not only making it highly complex, but also limiting the adaptability to other problems even further. On top of this fact, the interesting parts of such planners are often specific to the domain and problem type it was designed to solve, in this case the highly iterated upon heuristics, meaning it has little relevance to other planning instances. Therefore, while practical, it is an inappropriate test bed for integrating

emotions since any results would only apply to that specific planner.

2.4.2 Domain Independent Planners

Domain independent planners on the other hand are the main focus of the research world. The idea behind this type of planner is that it can work in any domain without modification. In reality, achieving this is extremely complex and a large number of restrictions and assumptions limit the feasibility of practical application. In a relatively simple scenario of planning for container stacking by harbor cranes, the complexity rapidly increases per extra container and crane added, to the point where a domain independent planner is impractical in most harbors in the world.

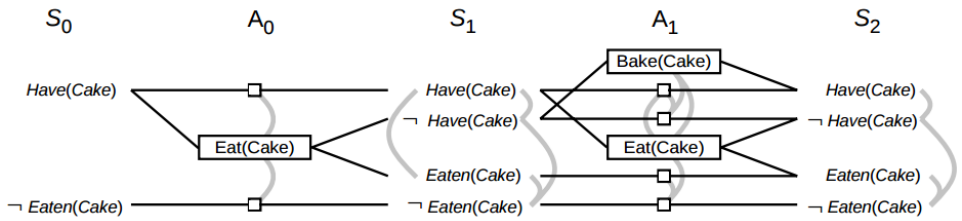
One often used method to be able to solve a scenario like the one mentioned above is to adapt the scenario to a classical planning environment. This means that the environments are fully observable, deterministic, finite, static, and discrete [30]. From here there are many planners that attempt to lift one or more of these restrictions in an attempt to to able to handle a greater number of domains and problems, however, these lifted restrictions often come paired with significant downsides such as significant increases in time or storage of said planners.

Graphplan is a well known planning algorithm that processes a planning graph using backwards search to generate a plan. First, the planning graph has to be created. A planning graph is a special data structure that consists of alternating levels containing literals and actions. The literals and actions present at a level are the ones that can be true or have their preconditions satisfied, respectively, at that level. The planning graph can be created incrementally, starting from the initial state. What this entails is that the next action level is created by looking at the previous literal level and if an actions preconditions are met then it is added to the next action level. The added action is then linked to its preconditions in the previous literal level and also linked to it's effects (literals) in the next literal level. This linking to the next literal level also adds literals to the next literal level in the process. Literals also have persistence to represent inaction, so each literal is added to the next literal level with a persistence action.

Lastly, the planning graph also has mutual exclusions. These are to prevent two conflicting actions or literals from occurring simultaneously, and are added in the form of mutex links between such actions and literals. A mutex relation exists between two actions if one of three conditions holds. First, inconsistent effects holds when one action negates the effects of the other. Second, interference holds when one of the effects of one action is the negation of the precondition of the other action. Third, competing needs holds when one of the preconditions of one action is mutually exclusive with a precondition of the other action. In the case of literals, there is a mutex relation if two literals at the same level are each others negation, or if inconsistent support holds, which is when each possible pair of actions that could achieve both literals is mutually exclusive. Figure 2.4 shows an example scenario in which a planning graph has been created for the 'have cake and eat it too' problem.

The actual graphplan algorithm consists of two steps that loop until a solution has been found [30]. The first step is to check if all the goal literals are present at the current level of the planning graph without any mutex

FIGURE 2.4: "The planning graph for the 'have cake and eat cake too' problem up to level S_2. Rectangles indicate actions (small squares indicate persistence actions) and straight lines indicate preconditions and effects. Mutex links are shown as curved gray lines." [30]



links between any pair of them. If this is the case a solution might exist and backwards search is applied to try and find it. The backwards search starts at the end/goal state and tries to find a sequence of actions that will bring it to the starting state. Otherwise, the second step will add another level to the graph using the incremental planning graph creation process described previously.

These two steps loop until a solution is found or it is determined that there is no solution. It has been proven, in a proof outside the scope of this explanation, that a planning graph eventually levels off. Therefore, it can be learned that there is no solution if at least one goal is missing, or at least two goals are mutex, and the next level of the planning graph is identical to the previous one.

These planners would be a much better test bed than the domain specific planners since integrating emotions with a domain independent planner could then be applied to a large variety of domains via this planner. However, graphplan is a simple domain-independent planner limited to classical planning problems, and the complex nature of more advanced and interesting versions of these kinds of planners makes integration with emotions a complex endeavor that would require deep knowledge and understanding of the domain independent planner involved. This would greatly limit the time that could be spent on investigating the possibilities for integrating emotions. Therefore, integrating with a domain independent planner would be more appropriate as a next step, after the best methods for integration have already been explored.

2.4.3 Domain Configurable Planners

Lastly, we have the domain configurable planners, which are a hybrid between the first two by allowing domain specific input to be provided to the domain independent planner. This helps to greatly reduce the number of possible states to chose from, which is what makes the domain independent planner unfeasible for complex practical applications. However, the domain specific information still has to be created for each new domain the planner must work with. The significance of the amount of work this takes was revealed when the lack of domain configurable planners taking part in recent planning competitions was questioned [24]. After the domain configurable planners had proven that they work very well, few were willing to create the new domain specific input needed for each competition. They

do however, show more promise in terms of being practical to implement when compared to domain independent planners.

Hierarchical task network (HTN) planners and control formula planners are two types of domain configurable planners that use domain specific input to reduce the problem to a state that the independent planner can handle efficiently. HTN planners decompose tasks into subtasks and enforce constraints on which tasks can be executed and when they can be executed. How to perform this decomposition of tasks is the domain specific input, and, for example, can be a method that decomposes a travel task into the subtasks of determining a mode of transportation and a traveling route. A constraint can then be made to determine that when the distance is very large, e.g. traveling from Amsterdam to New York, the mode of transportation should be an airplane. This task can then be further decomposed into the primitive tasks of buying a flight ticket, checking in, getting on the plane, etc. Once the primitive task level is reached and the goal is achieved without violating any of the constraints the planner knows it has created a successful plan.

Pyphop is a basic HTN planner based on the SHOP and SHOP2 HTN planning systems [23]. It's planning algorithm is similar to the one used in SHOP, but with some differences that make it friendlier to integrate with software. There are two main differences, the first is that Pyhop represents world states as variable bindings of a python object rather than logical propositions. The second is that the HTN operators and methods can be written as python functions instead of requiring a specialized planning language. A world state, for example, can be passed to a method as an argument.

In the case of Pyhop, methods are the functions that decompose tasks into subtasks. These subtasks are a set consisting of a combination of methods and operators. Operators are the primitive tasks, which means they are actions that can be directly executed. The methods are where the domain specific knowledge component is present. A method could, for example, be written to decompose a travel task into a set of subtasks to either walk or drive to the destination. Both of these subtasks then have specified preconditions, such as walking if the distance is less then five kilometers.

To find a solution a left-to-right backtracking algorithm searches through the tasks decomposing them into their respective subtasks as it encounters them. Whenever a method is encountered it is decomposed and the search continues with the resulting subtasks. If an operator is encountered the precondition is checked, and if successful the operator's action is added to the plan. Otherwise the search backtracks to the next method or operator and continues this process until either a solution is found or there are no more methods to decompose and no operators whose preconditions are met. If at this point no possible plan has been found that leads to the goal then no plan is possible.

Control formula planners on the other hand works based on rules such as "Never pick up x unless x needs to go on top of another block", in the case of a block stacking problem. These rules are the domain specific knowledge that can be provided to the planner, and using these rules it can prune the search space to a point where the independent planner is able to solve the problem efficiently. A tricky part of this type of planner is that the control

formulas should be derived from the previous state using logic progression, which is known to be a non-trivial task [15]. Since it isn't feasible to simply write out all the control rules as domain specific knowledge this type of planner requires a good deal more investment to get working when compared to HTN style planners.

2.5 Summary

The focus of the remainder of the thesis will be on the exploration of emotions, their function and effects, and how they can be integrated with domain configurable planners. In determining the function and effects of emotions we will take inspiration from the explored emotion models as well as papers that analyze these aspects of emotions in humans. For emotion-planning integration we will focus on three planning points of influence. These aspects are actions, task decomposition, and goals. We chose to domain configurable planners for testing scenarios as they are a suitably stable test bed where we have a high level of control over the planner's inputs without having to delve into a complex domain specific planner. As an added benefit the independent planner can theoretically be switched, meaning any integration techniques can be applied to a wide variety of planners.

Chapter 3

The Function and Effects of Emotions

3.1 The Influence of Emotions

Emotions were originally considered to be a negative and unpredictable influence on the logical decision making process. It has since been shown that emotions are actually a critical component of the decision making process [14]. There is research shedding light on how emotions affect people from their overall choice patterns, all the way down to their day to day decision making. Lerner et. al [17] have compiled a selection of what important discoveries have been made in the last few decades about emotions and decision making. From this, and other available sources like this, we attempt to sort the effects emotions have on decision making in order to facilitate the creation of an integration model for emotions and planning.

3.1.1 Modeling Emotions

For the discussion and analysis of the function and effects of emotions in humans we are going to categorize them into the basic and complex emotion categories. Basic emotions are those that govern our general decision making process. These are emotions that determine how we react to the world around us and are limited to the ones that play a role in almost all interactions. Complex emotions on the other hand are the emotions that govern our behavior on a social and long term level. This includes reactions to the actions of others but also to events that have lasting effects.

Another distinction we make for modeling emotions is how they affect behavior. This is not to say these emotions can only affect behavior in this way, it is mostly a guideline for their main method of interaction with our decision making process. In this sense we classify the discussed emotions in the following manner:

1. Enabling and disabling behaviors
    - Anger, fear, gratitude
2. Guidance and steering of the decision making process
    - Happiness, sadness, sympathy, pride, embarrassment, shame, guilt, jealousy
3. Triggering behaviors or adding goals

- Disgust, surprise, envy, admiration, indignation, contempt, anticipation

The first category consists of emotions that affect the decision making process by enriching or limiting the choices that can be made. This results in increasing or decreasing the number of possible actions while these emotions are present. It is possible, however, for these emotions to have little effect on the decision making process if the added or removed actions are not relevant to the current goals and plan.

While emotions of the first category add or remove potential actions while the emotion is present, the second category of emotions works on a more long-term scale. These emotions alter the planning strategy and goal prioritization. This type of change represents a core change in how decisions are made and is generally introduced gradually. Another aspect of such a change is that when the emotion subsides the effects can linger.

The third category contains emotions which cause an immediate reaction that can take priority and has to be integrated with the current plan. These emotions achieve this by triggering behavior or creating an additional goal for the planner to fulfill. This immediate reprioritization separates it from the first category. The effects of the first category can be minimal, however, these emotions have triggered behavior or added a goal that must now be taken into account.

3.1.2 Basic Emotions

First we look at the basic emotions as determined by Ekman: happiness, sadness, anger, disgust, fear, and surprise [9]. Even though these are considered basic emotions their effect on the decision making process, and therefore on a planner, are not. For example, while happiness is a positive emotion, it does not necessarily lead to better decision making. At the same time negative emotions such as disgust can help to avoid people or actions that can lead to a negative outcome. However, for most emotions there are identifiable and predictable effects when experiencing them.

Lyubomirsky et. al [20] have looked into the effects of long-term **happiness** in terms of success in a person's life. While it is hard to have conclusive evidence on such a question, they do point out some interesting discoveries. In general, positive emotions do seem to lead to more optimal decisions, and the same is true for negative emotions leading to less optimal decisions. This most likely stems from the fact that a happy person has a working decision making strategy that lead to the happiness in the first place. This means that the strategy can be optimized relatively safely, while a sad person would want to find a new strategy, which involves trying out new, and therefore risky, things. This leads to happiness providing a desire to stick to and improve the current strategy, reducing the emphasis on trying completely new things.

On the other side of the spectrum we have **sadness**, which influences in the opposite direction, encouraging experimentation in the hopes of finding a strategy that works and leads to happiness [32]. Due to the high risk reward nature of this behavior it often leads to changes that have a strong effect, significantly changing the situation, although not necessarily for the best.

Another influential emotion is **anger**. It has proven to be a very useful emotion that stems from being treated unfair or wrong, and forces decision making to focus on fixing this wrong. Anger allows someone to work against another person or obstacle with a sort of 'any means necessary' attitude [11]. This can be seen as a removal of normal limits on actions that can be taken to solve a situation. These limits come in many forms, such as social rules or risk reward evaluation. As mentioned anger is an overriding emotion, however, it can also override other emotions such as fear of reprisal.

There are also some emotions that at their core are easy to understand and describe. **Disgust** can stem from a plethora of causes, however, the end result is the expelling and avoidance of the source since it is now undesirable to be associated with it [28]. For **fear** there are a large variety of manifestations and reactions, yet the source can always be traced back to a goal (potentially) being violated [12]. Whether this goal was still to be achieved or maintained, it is this potential of loss that produces fear, and once this goal is no longer threatened the fear subsides.

Finally, **surprise** is an interesting emotion since its effect seems to be very nuanced [17]. When something unexpected happens we become surprised, which causes us to pay careful attention to what is happening since our previous prediction of what would happen was apparently wrong. This emotion can be seen as a wake up call that lets us know that our automated assumptions have failed and we have to properly evaluate the situation and update our failed assumptions. In this way surprise acts as a mechanism that forces us to pay careful attention to an unexpected situation.

3.1.3 Complex Emotions

Besides the basic emotions there is a wide range of acknowledged complex emotions, many of them play a key role in social interactions. Damasio distinguishes the emotions sympathy, embarrassment, shame, guilt, pride, jealousy, envy, gratitude, admiration, indignation, and contempt [5]. Together they provide the emotions needed to simulate human emotional reasoning in social scenarios. These emotions have a wide variety of effects, some backed up by research and others by experience, so we will have to make some assumptions about how we position these emotions in a model for integration with planners.

First lets look at two emotions that greatly enhance our ability to form and maintain social relationships. **Sympathy** is the ability to perceive, understand, and react to the needs of another [6]. By being able to recognize and act on this need we can seek to provide aid in solving their situation. An interesting part of sympathy is that it seems to be easily suppressed when deliberating on why one is feeling sympathetic [31]. For other emotions this suppression is far more difficult to achieve. Without sympathy the needs of another can easily be missed or ignored, potentially negatively affecting the relationship with this person.

Gratitude helps us reciprocate in a relationship, facilitating the maintenance of said relationship. On top of this gratitude also stimulates the ability to help a benefactor even if this is costly for one's self [2]. This even translates to larger systems, such as aiding a communal economic venture at the cost of one's individual financial gains [21]. This makes it one of

the more beneficial social emotions, allowing for the improvement of social relationships and constructs.

Besides emotions that make social relationships beneficial to engage in we also have emotions that aid in fixing mistakes, particularly in the case of socially unacceptable behavior. Embarrassment, shame and guilt are often confused but there are clear differences between them. **Embarrassment** stems from a situation where socially unacceptable behavior is witness by others [33]. It makes us worried about how others judge our actions, and simultaneously motivates us to not repeat the mistakes we made. However, embarrassment is something that can be looked back on as a positive learning experience.

Shame on the other hand is similar in motivating us to not repeat our mistakes, but is not something we want to remember. This is due to shame being caused by a comparison of one's actions to one's standards, meaning it stems from a more critical moral dilemma. It also does not rely on the observation by others as one can be ashamed without anyone knowing about the mistake that was made. Experiencing shame is therefore a very powerful tool to prevent mistakes that will have a negative impact on one's self or their social status.

Last in this trio we have **guilt**, which, while similar, is focused on having done something wrong to another. It also prevents the repetition of the mistakes that were made, however, it also motivates to make reparations. This difference makes it very important for maintaining healthy social relationships. It functions as a backup plan when mistakes have been made that can cause such a relationship to fail.

Pride is a difficult emotion to classify. It causes a person to have confidence or see value in one's choices and actions. However, this can be seen as a positive and negative effect. In the positive sense it can be a feeling of fulfillment and positive self-reflection [18]. However, others can see it as an irrational inflation of one's value or accomplishments. Socially, a person experiencing pride is seen as either having high social status in a group or the opposite, depending on how the other group members view them in the first place. This makes it a very peculiar emotion that works as a self-reinforcing mechanism for your own behavior, with the potential to backfire in social situations if others do not agree with the behavior that you are reinforcing.

Emotions can also enhance learning by putting emphasis on a role model. This emotion is known as **admiration**. This emotion has a very elegant function. Since humans have benefited greatly from learning from each other, this emotion is experienced when we believe that someone has knowledge that would be beneficial to learn, motivating us to approach and copy them [13]. The downside is that this emotion can also be experienced in cases where, especially from a social standpoint, it will have a negative effect if everyone was to adopt this behavior. Admiration is also more likely to arise from figures that already align with one's own view of the world, leading to a self-reinforcing cycle that promotes tunnel vision of a certain kind of behavior.

Next, we look at two emotions that focus on the self in a social context: jealousy and envy. Both invoke negative feelings towards another, however, the distinction between these two lies in the nature of the situation. **Jealousy** is motivated by the fear of losing something one already

has, while **envy** is characterized by the desire to obtain something you do not [25]. Jealousy will also produce feelings of distrust and anger towards the person causing the jealousy, while envy gives a sense of inferiority and resentment. From a practical standpoint they ensure that in a social context there are also emotions that facilitate putting one's self first.

Besides anger, there is a more complex reaction to being wronged. **Indignation**, besides being considered a long-term version of anger, is also one less prone to inducing aggressive behavior. It functions more as a motivator to right a wrong, generally without resorting to deplorable behavior. This is partly because indignation is often caused by someone performing actions that are against your moral values, but can also stem from a wrong that was not specifically inflicted on you [22]. Therefore it has the function of motivating people to react to others engaging in negative behavior, a sort of social control emotion.

Contempt is an emotion that generally results in the rejection and social-exclusion of the another person [10]. It is an emotion that stems from a negative judgment of a person due to some moral or personal failing of theirs. It motivates us to stay away from people that don't align with our morals or standards. On top of this it also motivates us to convince others to adopt this stance towards the source of our contempt. While this behavior can be beneficial in some cases it also leads to excluding behavior which can be detrimental to maintaining a healthy social system.

Lastly, **Anticipation** is an interesting emotion that motivates one to imagine what could be and how this would effect one's self. This reflective behavior can be used to make decisions to pursue or avoid certain actions or situations [3]. It is a very useful emotion that allows us to deal with complex situations, which might otherwise overwhelm us with too many uncertainties, by imagining the consequences of our actions. Anticipation can also emotionally soften a negative outcome because we have already prepared for it.

Chapter 4

Planning With Emotions

4.1 Emotions and Planning

The next step is to look at how these emotions can be incorporated with a planner. First we will look at the possibilities for influencing a planner. Allowing us to discover appropriate ways to implement an emotion's effects. After that we will look at applying what we have learned and discuss which mechanisms to use to simulate each emotion's effects.

4.1.1 Influencing A Planner

Since the number of available planners is vast we will attempt to identify generic planning components that align with the three categories of emotions that were identified in chapter 3. These categories are enabling and disabling behaviors, guidance and steering of the decision making process, and triggering or adding goals. By limiting ourselves to planning components that correspond with emotional functions we seek to find a natural synergy between emotions and planning similar to our own. Additionally, the emotional state and emotion reasoning is performed outside of the planner, in what we will refer to as the emotion module. This module can use one's own implementation or make use of an existing emotion framework to perform these tasks. Further details on this module will be presented in 5, for now it suffices to know of its existence.

Actions

Actions are a basic planning component that have potential for implementing the first category of emotions. Actions determine what an agent can and cannot do. Whether an action can be performed is generally based on preconditions. The result of a performed action is modeled as an effect or postcondition that changes the world state, the agent's own state, or the state of one or more other agents. By adding one or more emotional states to the preconditions of an action a planner can simulate modified behavior based on the emotions an agent is currently experiencing. E.g. aggressive actions with a precondition that the agent's emotional state includes anger. Adding emotional states to the effect (postcondition) of an action in turn allows an agent to change its emotional state based on its actions. E.g. taking a moment to collect one's self to relieve stress. An important part of this construction is that once an emotion is gone the agent can no longer perform the associated actions.

Since an emotion does not correspond to a static set of actions we need some sort of model for determining when actions should have a certain

emotion as their pre- or post-condition. Determining which actions are relevant for an emotion depends on a wide variety of factors, however, by using domain specific knowledge we can provide each action with relevance factors for every emotion. This link from emotion to its relevant actions creates a model that simplifies the process. The relevance factor can be represented by a value that determines how relevant an emotion is to this action. Each action would then have a relevance factor for each emotion that is relevant to it, otherwise it can be omitted or set to a neutral value. Emotions, which also have a strength, can then add actions to the planner whose relevance factor is met. This means that we no longer have to specify a separate action for each combination of emotions. Instead we add relevant actions based on the emotional state. Since there are most likely no or few actions that are affected by every single emotion and we represent a finite number of emotions, this relevance factor specification should add a manageable amount of overhead to the creation of actions. Which, in the case of domain specific planners, already have to be manually specified. We add these relevance factors when the actions themselves are specified, before the planner is started, since we want to alter the actual planner as little as possible.

As an example, imagine an action that causes the agent to yell at their target, demanding an object that was taken from it to be given back. For ease of use we mirror the strength range of emotions (zero to one hundred) for our relevance factors. This action is relevant for the emotions anger, jealousy, disgust, and contempt. Their respective relevance factors are set to 70, 50, 30, and 30. For all the other emotions the relevance factor is set to zero. Now, for our agent to be able to utilize this action, at least one of the relevance factors must be met. If our agent is very angry, with a strength value of 90, this action will be added to the planner because the strength of the anger emotion exceeds the relevance factor for anger, which is 70. On the other hand, if the agent has an emotional state with anger at strength 50 and jealousy at strength 30 then the action is not added as none of the relevance factors of the action are exceeded by emotions in the emotional state of the agent.

Task Decomposition

Task decomposition is a planning tool that enables a planner to resolve complex problems by decomposing them into smaller more manageable ones. This decomposition is usually guided by an algorithm or function, but it can also be done using preconditions. In the case of preconditions, a task is decomposed into its subtasks if its preconditions are met. The decomposition function or preconditions can be modified, which will lead to the planner to produce different task decompositions. Modifying this component provides an opportunity for high level influence on the planning, which matches the second category of emotions. An agent's ability and willingness to perform certain actions can become limited or enhanced by emotions. Different sub-tasks become available depending on the emotional state, which causes the planner to produce a plan that aligns with the emotional state. In most cases new actions and behaviors will have to be added in order to allow for a different decomposition to be possible. When the emotions that made changes fade, the related actions and behaviors are no longer executable.

Here we can make use of the previously mentioned relevance factors of actions. In a hierarchical task network (HTN) actions are the lowest level of task decomposition. Since, if there is a possible solution, a decomposition eventually has to reduce to the lowest level we can look at aggregated relevance factors to determine if a decomposition aligns with the agent's emotional state. This aggregate is determined by averaging the relevance factors for each emotion which exists at least once in the decomposition. This also means that we can determine the relevance factors at any decomposition level by aggregating the relevance factors of this decomposition's subtasks. The actual value of the threshold can also be scaled according to the relevant emotion's strength. For example, a very angry person will have a high threshold, meaning that the average anger relevance factors of the actions in a decomposition has to be very high in order for it to meet the threshold.

Since we merely link emotions with tasks, we do not increase the amount of possible subtask combinations. Instead we actually reduce the number of possible combinations since only decompositions that meet a similarity threshold are considered by the planner. This similarity threshold is based on how close the relevance values of each emotion of the decomposition are to the emotional state of the agent. The actual strictness of the similarity can be defined using domain-specific knowledge about, for example, the agent's emotional sensitivity and empathy. This aggregated relevance factor per emotion allows us to see how any given task decomposition aligns with the agent's emotional state. With correctly specified relevance factors we can prune the possible decompositions before feeding them to the planner. The planner is then only able to select decompositions that align with the emotional state. In order to prune the planning state we have to add a mechanism that will allow us to prune decompositions at each step of the planning. Since HTN planners calculate their plan iteratively we can step in between iterations and filter the decompositions according to the emotional state.

The general idea of this process is as follows. Prior to running the planning algorithm we must determine the average relevance factors for each subtask. This can be done in multiple ways but since the subtasks network is a tree one way is to perform a sort of depth first search for all of the actions. Since we are looking at a hierarchical task network we know that there will be few top level tasks when compared to actions, motivating us to use a top down search. Starting at the highest level subtasks we step down into each subtask until we run into an action at the bottom level. This action's relevance factors are then added to every subtask that was passed along the way down the tree. We repeat this for every top level task. Once the depth first search has found all the actions each subtask has an aggregated list of relevance factors for all its potential actions. We then average these relevance factors resulting in an aggregated average for each subtask. Now that we have the average relevance factors we simply filter out the subtasks whose relevance factors do not meet the decomposition thresholds at each iteration of the HTN planning algorithm, meaning the planning algorithm only gets to select subtasks that align with the current emotional state.

Goals

The third category of emotions relates closely to planning goals. A planning goal is usually represented as a state. It can include specifications of the world state, an agent's own state, or even the states of other agents. The planner then makes use of its other planning components to determine one or more sequences of actions that result in the goal's state being met. We will limit ourselves to the assumption that all goals are of the same type, a state which the planner has to satisfy. Multiple goals will be interpreted by the planner as an aggregation of the states of these goals, meaning that all goals must be met simultaneously in order to find a solution. As a result of this, conflicting goals will prevent the planner from finding a solution. The upside of this limitation is that what constitutes a goal is well defined and this model will work for almost any planner. For the sake of goal manipulation goals are kept separate until fed into the planner.

Emotions such as disgust, which can cause an agent to expel any objects that it is disgusted with, translate particularly well to adding an extra goal in which the agent no longer possess any objects it is disgusted with. Emotions that trigger behavior can also be simulated using goals by forcing the planner to converge to a state where this behavior's postcondition is present. Similarly to the other categories, meeting such a new goal could benefit from new actions being added to the planner.

Since we do not have access to priority mechanisms or different types of goals it is more difficult to simulate the fading strength of a goal's relevant emotions. We cannot simply lower a goals priority to reduce a planner's eagerness to fulfill the goal. However, the prevention of goal removal is important because it facilitates the handling of long-term emotions. Even though emotions eventually wane in strength, action might still be required to fully resolve the situation that caused the emotion in the first place. To be able to simulate this effect we can attempt the planning with the goal included, if the planning is unsuccessful we can redo the planning, removing goals whose relevant emotions (relevance factors) are below a certain threshold. This can, however, have a significant impact on performance and it might be better to limit the re-planning or even omit it entirely. The goal threshold value is specified for each emotion. This is because emotions differ in when their strength is low enough to warrant goal removal. At the same time not every goal requires their own threshold since emotion motivated goal removal should be based on each agent's individual sensitivity to each emotion rather than each individual goal's specifics. In the case of no re-planning, goals will be removed once the relevance factors of its emotions fall below a threshold. The idea behind this implementation is that the emotions of a goal will only fade if the source of these emotions is removed, whether this is due to an agent acting on a goal or otherwise is not important for determining goal removal. One side effect of using thresholds for goal removal is that goal activation and removal can potentially oscillate around the threshold, leading to rapid adding and removing of those goals. However, this behavior is not detrimental since something is still pushing the goal's relevant emotions over the threshold, which should cause an emotional response.

As mentioned, domain configurable planners are an interesting case. They allow for domain specific input that can completely change the way

the planning process plays out. As an added benefit, proper specification of emotion effects and implementation guidelines can facilitate the creation of domain specific input for any domain configurable planner. Since this input is utilized to prune the potential states the planner has to explore, each emotion could have its own a control function which prunes states that do not align with the emotion currently being experienced by the agent. Indignation towards an agent could cause states that benefit this agent to be pruned.

Due to the complexity of simulating realistic reactions, domain specific knowledge will be crucial for the implementation of many of the emotions discussed below. To avoid repetition of this requirement, and because we see this knowledge specification as a tool to be used where needed, it will not always be mentioned when an emotion requires such extra knowledge.

Model Summary

In summary the model adheres to the following rules:

- Every action has a relevance factor specified for each emotion (can be omitted or a neutral value if the emotion is not relevant)
  - Actions that have a target must be able to receive this target as a parameter
- Task decompositions are filtered according to a threshold value that is evaluated against the average relevance factors of a decomposition
- Goals have their own threshold value per emotion that, if met, adds the goal to the planner's goal state, otherwise it is removed

By implementing these rules the model provides a framework for anyone wanting to integrate an existing automated planner and emotion system. As we will explore in the rest of this chapter the actual values of the relevance factors and thresholds have to be determined separately. As in, the model does not provide an automated way to determine an appropriate threshold for a certain action or goal. In the next chapter we will explore what the model adds in terms of enabling the communication between the planner and emotional system.

4.2 Combining Emotions and Planning

In this section we will take a look at how each of the basic and complex emotions can potentially be implemented. We base these potential implementations on the components explored in the previous section, giving a small example of how this implementation would work out in practice. The next chapter we will focus on a specific implementation methodology.

4.2.1 Basic Emotions

Simulating the effects of **happiness** on planning directly might not be necessary. An emotional state of happiness means that things are going well, and therefore, the planner's behavior doesn't have to be altered. For planning purposes happiness could be seen as a neutral state where the current

planning method or strategy is satisfactory. Another option for simulating happiness could be via a threshold that prevents changes to the current task decomposition function. This would represent the fact that the current planning has in some way lead to happiness, which is a desirable emotional state. This embodies the philosophy of "if it isn't broken don't fix it".

In the case of **sadness**, it does have a distinct effect on planning. Since the current situation is undesirable, change is needed. One way to accomplish this is to allow for actions whose relevance factor has been met to be enabled when in this state. However, since sadness has a more pronounced high level effect, altering the task decomposition function could allow for a true shift in the planning behavior. By relaxing the decomposition thresholds for all the emotions the planner will have more actions to choose from. In combination with the newly enabled and relevant actions this will simulate the desire to change the current state of affairs.

If, for example, an agent has become sad due to music they are listening, it will cause any actions that have been specified as sad behavior to be added to the planner. At the same time the relevance threshold is relaxed, meaning that more decompositions are now possible. The planner can then choose to stop listening to the sad music, a newly added action, and proceed to go outside and meet up with some friends, an existing action that did not meet the threshold before since the agent was fine with listening to music.

Anger is an emotion characterized by significant behavior changes, making it a good candidate for adding and removing potential actions. These new options could allow the planner to resolve previously unsolvable situations. Or, due to the addition and removal being based on their emotion relevance, change the set of possible actions to more closely reflect the emotional state. Another component of anger is its ability to overtake the decision making process. This can be simulated by increasing the anger threshold for decompositions, only allowing decompositions that align strongly with the anger emotion. With the new actions and the threshold changes the agent's behavior will be significantly more aggressive. Lastly, anger generally has a target towards which it is directed. This means that anger should focus actions towards this target. When the actions are being specified this target component can be included as a parameter that is passed to the function which performs the actual action, allowing the function to apply this action to the target.

This emotions implementation works out very similarly to sadness, however, this time the possible decompositions have to align more with the emotional state as the agent is angered more. This will result in anger relevant emotions making up the bulk of behaviors. However, a decomposition does not have to exists purely out of angry behavior since the threshold is based on the average relevance factors of a decomposition and not on a per action basis. After all an angry person does not lose complete control over his decisions until they are truly enraged, at which point the threshold will reflect this by only being possible to be met with decompositions consisting of purely angry behavior.

As an example, imagine our agent is a toddler named Jimmy playing with his favorite toy. When another toddler, Steve, steals his toy it causes Jimmy to become very angry. It is his favorite toy and therefore he is so angry that the task decomposition threshold for anger also becomes very

high. This means that the next chosen task's decomposition must consists of actions that meet this threshold. We have specified several tasks that Jimmy can perform. He can ask if he can have his toy back (no anger relevance), demand it back (medium anger relevance), or forcibly take it back (high anger relevance). Due to Jimmy's high level of anger the only choice that meets the decomposition threshold is to forcibly take it back. This tasks consists of several high anger actions such as grabbing the object while it is still held by Steve, wrestling it from Steve's control, or even attacking Steve. If the toy was less important to Jimmy he might not have become so angry and therefore meet his anger task decomposition threshold by choosing one of the other tasks.

Disgust is a protective emotion in the sense that it helps to avoid people or objects that are perceived negatively. Disgust towards something can be represented by adding a goal for avoidance of the source of disgust. This goal, in combination with relevant actions, then forces the planner to incorporate this avoidance behavior. Similarly to anger, the source of the emotion will have to be provided to the related actions so that the actions know which objects have to be avoided or expelled.

As an example, when someone discovers that their favorite fast food company takes part in some negative business practices they can become disgusted with the company. This will cause a goal to be added with the company as its target. Since the planner tries to satisfy all the goals, and this goal prevents the company from being part of the plan, the avoidance behavior is realized. The disgust emotion is also strong enough to meet the relevance threshold for the expel objects action. This action expels any objects that are provided to it, in this case the company. Which objects would exactly have to be expelled should be specified as domain specific knowledge as this is very different behavior in regards to a company or a person. Only when the disgust emotion's strength has faded, eventually dropping below the goal's threshold, will it be removed. The same happens for the action, only in this case the action's relevance factor is compared to the disgust strength. At this point the person can once again interact with the fast food company.

Fear is an important emotion that emphasizes our desire to survive. From the decision making perspective, fear leads to avoidance behavior in regards to the feared people or things. In cases of extreme fear this is achieved in the same way as disgust avoidance, by adding a goal that requires the source of fear to be avoided. Similarly, we can also remove actions that lead to interaction with the source. In more mild cases the avoidance of confrontation with the source is more appropriate. However, such nuisance would have to be created using domain specific knowledge. For example, allowing for a wide variety of avoidance actions/goals ranging from complete avoidance of source and anything related to only avoiding direct confrontation with the source.

Another well know reaction to fear is the fight or flight response. This behavior forces a quick decision to be made. Depending on the state of the agent a goal can be added which enforces either fight or flight behavior. With two corresponding sets of actions that can only be chosen if their response type goal is present, it will force the planner to create a plan that includes behavior depending on which type of the fight or flight goal is present. The fight and flight goals can be specified with a fear threshold

that cannot be equal to each other. When only one of the thresholds is met that goal is added. When both thresholds are met the one with the highest value takes priority, similarly to how the action with the highest satisfied emotion relevance threshold is chosen.

For example, our agent runs into Casper the friendly ghost. While our agent is looking at a ghost this one is not particularly scary. The mild fear causes an avoidance goal to be added. However, this one has a low threshold and merely prevents the agent from physically interacting with the ghost (just in case this is possible). The planner therefore chooses from actions such as stare at, ask a question, check if dreaming. All actions that do not conflict with the avoidance goal. At this point the agent notices that the Ghostly Trio, not so friendly ghosts, had appeared behind him. This causes the agent to experience a high level of fear, causing both the fight and flight goal thresholds to be met. For our agent the threshold for the flight goal is the larger of the two. The planner, which due to task composition thresholds now has to choose high fear relevance actions, has access to fear actions such as scream, fetal position, and faint. However, it also has gained access to the run away as (fast as possible) action due to the flight goal being activated. Having to also satisfy the decomposition goal and the high fear relevance of the run away action, the planner chooses this action over the others.

Lastly we have **surprise**, which is an interesting emotion that arises when things have not gone as expected. It is basically an alert that lets the planner know that that the current plan needs to be reevaluated in order to account for unexpected events. In planning terms this translates to "restarting" the planner. However, since the planner cannot know when unexpected events will occur during the execution of a plan, this "restart" has to be an outside trigger. This is partly because we are looking at planners that create a complete plan before execution starts (off-line planning). The mechanism to force a re-planning is outside the scope of the currently discussed planning loop. However, implementing such a mechanism should be trivial as it simply requires a new call to the planner with the updated state, with the trigger for the call being based on how surprised the agent is.

4.2.2 Complex Emotions

There are several components that have to be taken into account to accurately simulate **sympathy**. Firstly, sympathy is a targeted emotion and therefore the relevant actions have to be provided with a target, the source of the sympathy. Second, the general effect of sympathy on the task decomposition will be to favor tasks that benefit the source of the sympathy. This is made possible via the use of a decomposition threshold which enforces a minimum average value of sympathy relevant actions to be included in the task decomposition. Lastly, the sympathy relevant actions are made available to the planner based on their sympathy relevance factors. The details of these actions and their relevance factors should be defined using domain specific knowledge.

For an example of how this would work we can look at the anger example. While these are very different emotions the implementation using relevant actions and forcing those actions to be used via the task decomposition

threshold value is the same. Anger and sympathy also share the aspect of being a targeted emotion. The ability to implement many emotions in the same manner is part of keeping the model generically applicable, not only in terms of planners but also emotions.

The effect of **gratitude** is very similar to sympathy and, for the purposes of our implementation, is virtually the same. The main difference is that gratitude actions can also include behavior that negatively affects our agent. This behavior, however, is specified in the actions themselves using domain specific knowledge, which means we still handle the emotions and where they influence the planning in the same fashion. Actions whose gratitude relevance factor is met are added to the list of available actions and have to be specified with a target parameter. At the same time the decomposition function has a gratitude threshold value which a task decomposition has to meet in order to be a valid decomposition.

In terms of a basic planning implementation **embarrassment**, **shame**, and **guilt** are very similar. They each originate due to actions that result in negative feedback and motivate the agent to not repeat these mistakes. This is implemented by using a decomposition threshold value for each of these three emotions. However, in this case the threshold value is used as an upper bound. This means that a decomposition cannot contain too many relevant actions as this would cause the decomposition's average relevance factor to exceed this threshold. This prevents mistakes from being made again since actions that cause negative feedback in the form of these three emotions are now, proportionally to the emotion's strength, avoided by the planner.

Embarrassment, shame, and guilt are, however, applicable to different variations of severity and have their own nuances in terms of resulting behavior. Embarrassment is the mildest version of negative emotional feedback, while shame is generally more severe due to stemming from a moral failing. This can be reflected in the strictness of the produced relevance factors and decomposition threshold values. Guilt has the additional component of wanting to help the person that they wronged. This is implemented in the same manner as sympathy and gratitude; making use of a decomposition threshold value and guilt relevant actions that focus on making reparations.

For example, Tom has said spread some rumors about his friend Jessica. Bob, a friend of Jessica, discovers that it was Tom who spread these rumors and together they confront Tom. As they were rather vicious rumors and Tom is confronted by the duo he starts to feel very ashamed. Seeing how upset Jessica is he also feels guilty. The spreading rumors action has a high shame and guilt threshold. Since the task decomposition thresholds for shame and guilt cannot be exceeded by the selected actions, Tom can no longer continue to spread rumors about Jessica. At the same time his guilt causes guilt relevant actions to become available. Tom's guilt causes him to choose a method that tries to fix his mistakes, consisting of the actions apologize to Jessica and let everyone know that he was spreading false rumors.

Pride is an emotion that provides positive self reinforcement of one's behavior. Integrating this behavior into a planner can be handled in with the same methods as emotions such as anger. Pride relevant actions become available as the pride relevance threshold for these actions is met. A pride decomposition threshold value also has to be taken into account for

a decomposition a valid planner choice. The potential negative social consequences that can result from pride do not have to be handled here, and will instead be covered naturally by emotions such as embarrassment and shame. Since pride can be implemented with the exact same mechanisms as anger and sympathy one can look at their examples for details on the inner workings.

The learning of new behaviors is facilitated by **admiration**. Unlike most of the other emotions so far this requires some special functionality in order to achieve. Our agent needs to be able to either adopt actions from the admired agent, or be able to observe and copy their actions. These actions are then made available to the agent based on their admiration relevance factor, which is determine by the strength of the admiration emotion. With such a mechanism in place we can once again make use of the decomposition threshold value in order to filter task decompositions down to only the emotionally relevant ones.

As an example we can look at agent Johnny, who loves action movies with as many explosions as possible. Unsurprisingly he is a big fan of Michael Bay's movies and wants to become a filmmaker just like him. This translates to a strong feeling of admiration for Michael Bay and his movies. For this example we have admiration actions that mimic Michael Bay's actions. Johnny is a true fan and therefore his admiration allows the planner to choose many admiration actions: film a short action movie (low admiration relevance), try to improve car chase scenes with own editing (medium admiration relevance), or create special effects and explosions (high admiration relevance). All of these actions, with varying admiration thresholds are met by Johny's admiration. The planner chooses the strongest one, creating special effects and explosions. As Johnny works on this task he suddenly starts to become disinterested with his activities and his admiration decreases rapidly. Now the only choice left is filming a short action movie. However, he is also confused and angered by his sudden loss of interest in his hero. His anger emotion is now stronger then his admiration, and due to the anger decomposition threshold needing to be met the anger action of deleting his work is chosen instead.

Jealousy is experienced when there is a chance that one might lose something or someone they care about. Jealousy is a powerful emotion that can cause a variety of reactions. Anger and distrust towards the threatening agent will be handled by the emotion system itself. Through these new emotions jealousy already indirectly influences the planning. However, jealousy also has its own relevant actions which lean towards defensive behavior that attempts to negate the threatening situation. A jealousy decomposition threshold will then filter the task decompositions for those that utilize enough jealously relevant actions. At the same time the anger, a side product of the jealousy, will also be filtering the task decomposition, meaning there is potential for jealous and angry behavior. In terms of inner workings jealously can also be implemented the same way as anger.

Envy on the other hand will motivate an agent to acquire something that it does not have. Similarly to jealousy, envy produces some new emotions. In this case resentment and a sense of inferiority, which are again handled by the emotion system, enabling indirect influence on the planning. In terms of envy's effect on behavior, envy adds a goal which, together with actions whose envy relevance factor has been met, seeks to acquire what is

envied. A basic version of this goal can suffice in some cases, but for more realistic behavior the nature the source of envy should produce different goals. In a scenario where one is envious of someone's car the produced goal should be very different from envying another's physical appearance, as one acquires a car in a very different manner than one acquires another's physical appearance. At the same time people vary greatly in what they envy, therefore the best option is to use domain specific knowledge to define proper envy goals. A basic version of the goal can be implemented by containing a state in which the envied source is in the agent's possession. The implementation of envy is very similar to that of disgust, as both make use of a goal to enforce behavior in combination with relevance actions. It is only the content of the goal and actions that are different.

Even though **indignation** shares plenty of aspects with anger it should be implemented in a different fashion. Instead of enabling aggressive behaviors, it creates a goal with the intention to fix the wrong that has been done. For example, someone was purposely excluded from receiving a reward they earned. The goal would then contain the state in which they receive their reward. With the inclusion of indignation relevant actions this goal forces the planner to fix the wrong that caused the indignation in the first place. As we are making use of indignation relevant actions rather than anger relevant actions the behavior will be significantly different from anger inspired behavior. Again the goal and actions have to be tailored appropriately and the model will benefit from having multiple goals and actions with varying relevance thresholds that match the strength of the indignation emotion. In terms of implementation indignation can be implemented in the same fashion as disgust, which also uses a goal to specify a desired state and emotion related actions.

In terms of the behavior that results from **contempt**, it is very similar to disgust. However, in this case the behavior is the result of a personal or moral failing of another. A goal to avoid and exclude the source of contempt can be added, which will result in the planner attempting to reach a state where the source is not part of the plan. This goal can be achieved by once again making use of contempt relevant actions. As an additional component of contempt the goal should also include some form of excluding the source from the agent's social circle. The specification of such a goal depends on the underlying systems and is therefore done with domain specific knowledge. As mentioned this emotion is very similar to disgust and as such the implementation via a goal and actions is the same as for disgust and indignation.

Finally, **anticipation** can be realized by adding a goal that lets the planner know that certain things are expected to occur. This means that we can specify a goal that contains the state after these expected events have occurred. This anticipation goal causes the planner to create a plan for the anticipated world state, and if a plan is possible it allows the planner to handle a situation where the anticipated change does occur. Unlike most other emotions, anticipation does not have its own emotion specific actions. Instead the normal state is augmented with the anticipated state, meaning it has different emotional state, which in turn results in different emotion relevant actions, decompositions, and goals to be available to the planner.

For example, agent Timmy is a student living in a student house where they have a weekday schedule for cooking dinner. Today is Tuesday and

therefore it is his flatmate Bob's turn to make dinner. Timmy had to work late to finish a project and anticipates dinner being ready when he finally gets home. An anticipation goal is added which contains the state in which dinner is ready. Timmy plans his return trip and arrival around this augmented state. He doesn't need to pass by the supermarket, as food will already be available, and when he gets home he can drop off his stuff in his room and immediately head over to the kitchen to join for dinner. Thanks to the anticipated situation Timmy could realize a more streamlined plan.

Chapter 5

Representing Emotions and Their Effects

5.1 Representing Emotions

In this section we will take a look at how to represent emotions in a manner that facilitates implementation, providing guidelines on how to define emotions and manage the emotional state. Then in section 5.2 we present an agent model for realizing an integration implementation.

5.1.1 Components of an Emotion

In order to effectively represent an emotion we have to determine what information is needed to simulate it. Emotions are a reaction to an event or action. Therefore we want as much information about the emotion to determine when an emotion is relevant and should be generated. The components of an emotion are the properties that define the emotion. These are properties such as an emotion's presence, strength, compatibility with other emotions, the target of the emotion, and the source of the emotion. By specifying such properties we can make more accurate predictions about emotion changes and their resulting actions.

Preferably we want to be able to write a generic algorithm that can analyze the situation and figure out which emotions should be activated. However, this is very complex and outside the scope of this thesis. Instead we will make use of domain specific knowledge to make these decisions beforehand. For example, if a friend takes a bite from your hamburger, your reaction will be different very different from a situation where a stranger performs the same action.

In addition to an emotion's properties we also have to pay attention to the reason the emotion exists. For many emotions they can have either, or both, a source and target. The source is the reason the emotion was generated in the first place. There are many emotions that maintain their strength while the source is present. For these emotions, knowing what the source is can enable behavior where the emotion only starts to dissipate after the source is gone. The same idea works for the target of an emotion. When someone is angry due to a certain action or event, the person is likely to stay angry until the action is ceased or the event is finished. Therefore, tracking the source and target of an emotion can benefit the accuracy with which an emotion system can simulate emotions.

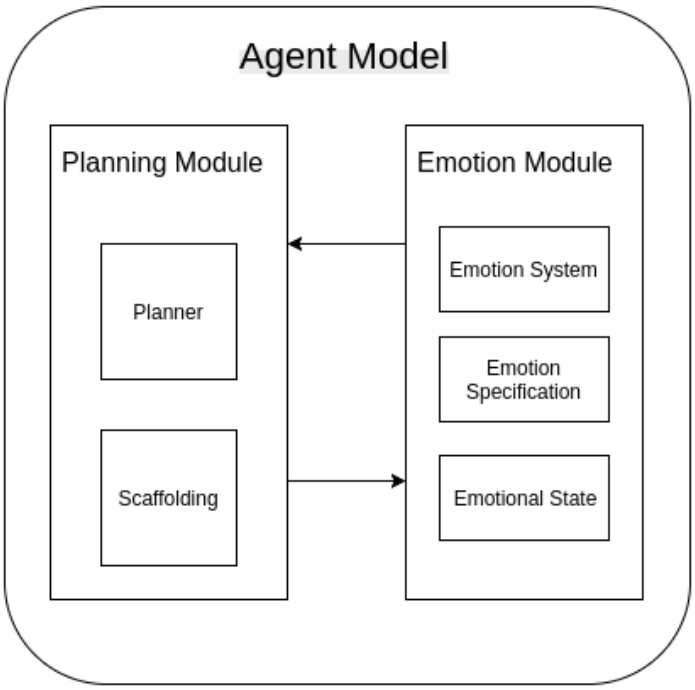
5.1.2 The Emotional State

Another topic is the representation of the entire emotional state, which can include multiple emotions at the same time. As described by Plutchik's wheel of emotions, there are emotions that conflict with each other [26]. To prevent odd situations where a person is happy and sad at the same time, one of the conflicting emotions must be pruned from the emotional state. There is no agreed upon list for which emotions are conflicting, but changing which emotions are conflicting is a domain specific issue that can be easily solved on a case by case basis. After specifying which emotions are considered conflicting, a method needs to exists to resolve these conflicts. A basic way would be to simply not add a new emotion if it conflicts with any of the existing ones. However, another, more elegant, option is to remove the conflicting emotion with the weaker strength. This allows the stronger emotion to triumph, while simultaneously mediating oscillating behavior.

5.2 Agent Model

The agent model that will be used to represent an agent that can plan using cognitive and emotion based reasoning consists of two modules: the emotion module and the planning module. The interaction between these two modules is described in the planning and emotion interaction section. Figure 5.1 shows a basic diagram of the agent model that will be discussed in the next two sections.

FIGURE 5.1: The agent model.



5.2.1 Emotion Module

The emotion module contains several important components. It contains the emotion system, emotion specifications, and manages the emotional state.

The emotion system is the main element of the emotion module. There are two options for this system. We can create our own emotion logic, allowing us to fine tune the emotional reactions in as much detail as we want, or we can utilize an existing emotion simulation model. By making use of an existing model we can choose one that is appropriate for our situation while also benefiting from the research and development that has gone into the model. The emotion system will receive the planning related input from the planning module while also having access to the emotional state of the agent.

In order to properly reason with emotions we need to maintain the emotional state of our agent. This state is updated according to the emotional responses generated by the emotion system. The state is represented by a collection of emotion objects (one for each emotion). These emotions can have a variable amount of properties where the emotion strength is the only mandatory one. If the chosen emotion system can handle other properties, such as an agent's susceptibility towards the emotion, these can be added as additional properties of the emotion objects. The collection contains one object per emotion.

Since the idea behind our model is to feed filtered actions and tasks into the planner we also have to maintain and specify the potential actions and their emotion relevance factors. This is called the emotion specification component and it contains all the potential actions and tasks that the planner can utilize to create a plan. This is where we add the relevance factors for actions, tasks, and goals. It is also where we filter the possible decompositions according to the emotional state. The resulting filtered set of actions, tasks, and goals are then send to the planning module.

5.2.2 Planning Module

In the planning module we find the planner together with the necessary components to properly format planner input and output for use in the emotion module.

The main component in this module is the planner. This can, similarly to the emotion system be a planner made according to one's own specifications, however it is desirable to use an existing planner in order to make use of the benefits that such a tried and tested planner can provide. The main restriction on planner choice is that it must be compatible with actions, tasks, and goals. This is because we use these aspects to integrate the emotions with the planner. Most planners make use of actions, tasks, and goals so the planner choice is fairly generous. An example of a compatible planner is Pyhop, a hierarchical task network that iteratively finds a solution using the provided actions, tasks, and goals. Hierarchical task network planners in particular, are a good choice since they align with the strategy used for the integration model.

In addition to the planner there are also some small extra components that depend on what extra formatting is required to feed the actions, tasks,

and goals to the planner. In the case of Pyhop, the iterative nature means we, for example, do no have to inject actions into the planner's state between planning steps. Instead we can filter the actions between iterations. Other components are related to extracting the planning state in a format that is compatible with the emotion module. This is usually required to enforce a consistent format in the planning information, allowing the emotion module to correctly and efficiently interpret this information.

5.3 Planning and Emotion Interaction Model

In this section we will describe how the emotion and planning modules communicate in order to enhance the planner with emotional reasoning. First we will give a quick overview of the communication steps, after which we delve into the details of each of these steps. Lastly we give an example of a single iteration (planning step) and the interactions that occur between the emotion and planning module during this process.

5.3.1 Model Overview

The process starts with an initial state from which a plan has to be created that reaches the goal state. This state is provided to the emotion system, which returns the initial emotional state. The emotion module then updates the emotional state according to the emotion changes and afterwards filters the set of available actions, tasks, and goals based on the new emotional state. Next, the filtered set is provided to the planning module, which uses the scaffolding components to feed the set into the planner. The planner executes a planning step and outputs the plan, which the planning module then forwards to the emotion module. Now the process repeats as this new state is fed into the emotion system.

5.3.2 The Process

The first step of every iteration is feeding the current planning state into the emotion system. This will allow the emotion system to determine which emotion changes occur due the planning decisions. The emotion system then outputs these changes, if any, which prompts the emotional state to be updated as necessary. After the emotional state is successfully updated the filtering process can begin. As specified in chapter 4 the set of actions and tasks are filtered by comparing their relevance factors to the strength of their respective emotions in the emotional state. Next, the goal state is modified according to the goal relevance factors. Once done the goals are aggregated into a single goal state. The set of filtered actions and tasks, together with the modified goal state, are send to the planning module.

At this point the planning module makes sure the information is in the proper format for being fed to the planner. If this is not the case the scaffolding components are used to properly format the information. It is then fed into the planner which performs a single iteration step. After the completion of the iteration step the planner outputs the new state. This state is then passed back to the emotion module where the entire process repeats until the planner outputs that it has completed all its iterations. Note that this model assumes the planner knows when it is done. If this is not the case

an extra component must be created to either determine when the planning is done or simply limited the number of iterations until failure is declared.

5.3.3 Single Planning Step Example

Lastly, we provide an example scenario in which one iteration step is performed. We keep this scenario very basic as to not clutter the description with excessive details. The starting state for our example scenario starts just after the planner has decided the agent should perform the apologize action. This action was selected because the emotional state contained a strong guilt emotion that met the apologize action's guilt relevance threshold. This action produces a state change, and this modified state is output by the planner. The new state is fed to the emotion system, which concludes that the agent is no longer feeling guilty and is happy about fixing it's mistake. The emotion changes are then processed by the emotional state component. The strength of the guilt emotion is lowered to neutral and the happy emotion is, as provided by the emotion system, increased to thirty. Now that the emotional state has been updated the filtering component is called. It starts comparing actions and tasks to the emotional state, which is neutral for all emotions except happiness. This causes it to filter out all actions whose relevance factor for happiness is less then thirty. Next, it determines which goals are to be added or removed by comparing the goal relevance thresholds to the emotional state of the agent. The goal to reduce the agent's guilt is removed since its threshold was twenty and the guilt experienced by the agent is now zero. Another potential goal, to high five another agent, happens to have a happiness threshold of twenty-five. Since the happiness of the agent is thirty, it exceeds this threshold and the goal is added to the goal state. No other goals are currently present and therefore the set of filtered actions and tasks, together with the aggregated goal state, are send to the planning module.

Our planning module accepts a set of actions, tasks, and a single goal. We already have our single aggregated goal state, however, the set of filtered actions and tasks must be separated for this planner. After converting the set into two filtered sets, one containing all the actions and the other all the tasks, we feed the sets and goal into the planner. After a single iteration the planner outputs the plan until this point. However, we want to know the changes, so that we can feed these into the emotion system. We make use of a scaffolding component created beforehand to determine the changes between the previous plan and the new plan. The resulting changes are then send to the emotion module were it is once again fed into the emotion system, completing our single iteration example.

Chapter 6

Experiments and Results

6.1 The Prototype

In this chapter we will put the model, presented in the previous chapter, to the test. To facilitate testing we have implemented a prototype that follows the specifications of the model. Most importantly, we integrate the emotions into an existing planner by creating an emotion and planning module together with the appropriate scaffolding components. By running several scenario's using this prototype, and observing the results with and without emotions, we attempt to validate the effectiveness of the integration facilitated by the model.

First we will discuss the planner that is used in the prototype, providing background on how it works and why it was chosen. Next we elaborate on how the emotions are implemented. Besides motivating the choice to manually create a basic emotion system for testing, we also tackle some of the implementation details that have to be taken into account when actually implementing the emotion module. We follow up with a discussion of the communication between the planning and emotion module. This also includes a global overview of the algorithm that the prototype uses to go from the initial state and goal specification to a plan.

After the explanation of the prototype we move on to testing. First we cover the scenarios that will be run using the prototype. Not only elaborating on the relevance and importance of these scenarios and their outcomes, but also an in-depth example of how the system proceeds to solve each scenario. Lastly, we conclude with a recap and evaluation of the successes and shortcomings of the prototype while also relating it all back to the model.

6.1.1 The planner

For the planner we are using Pyhop [23], a simple hierarchical task network (HTN) planner, written in python, by Dana S. Nau. It was chosen because it encapsulates the idea of the domain-configurable planner, while remaining simple and robust. This makes it a good fit for testing the interaction between planning and emotions without the planner's complexity biasing the results. At the same time it is also much more manageable as compared to, for example, epistemic planners, which due to their complexity would make monitoring the tests a far more convoluted affair.

Pyhop determines a plan by finding a sequence of operators that transforms the initial state into the state specified by a task. Operators are the most basic action and are the part of the plan that is actually executed by the agent. One level above this we have methods, which are compounds made up of one or more sub-collections of methods and/or operators. What this

means is that when a method is selected by the planner it is distilled into its relevant methods and operators. As mentioned, a task is basically a goal state that is to be reached, however, methods are assigned to specific tasks and multiple methods can be assigned to the same task. Which method is chosen for a task depends on the preconditions of the methods. One thing to note is that the actual planner only contains the planning algorithm and state. The rest of the components are completely separated and fed to the planner when run. This means that we can define and alter the operators, methods, and tasks outside of the planner. We make use of this by allowing these components to interact with the emotion module, for example, calling a method inside an operator that allows it to have an effect on the emotional state.

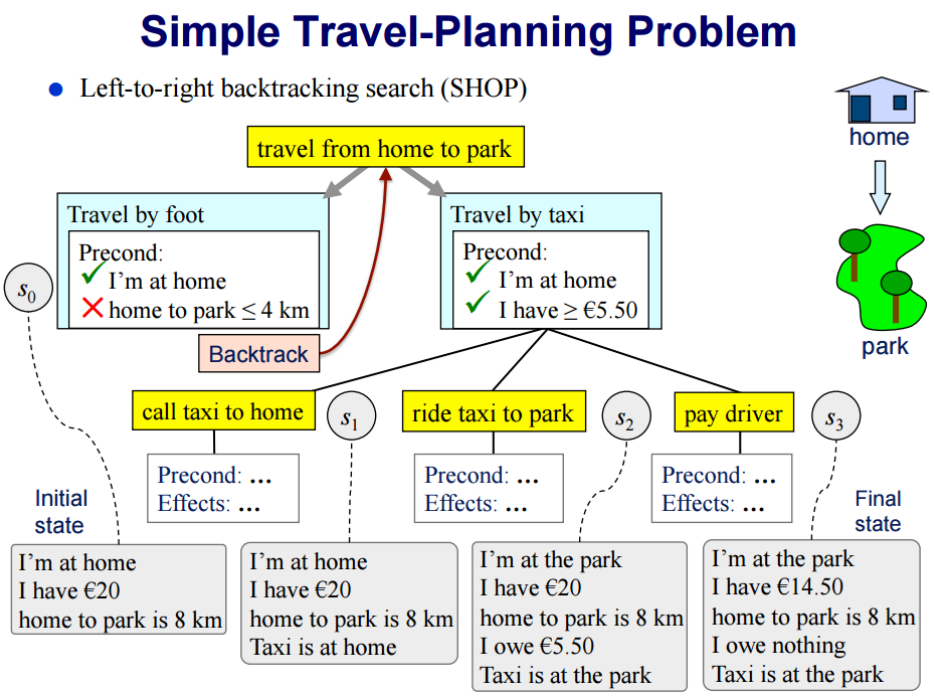
The algorithm for Phyhop boils down to two steps that are recursively executed until the task (goal) is completed or no more methods or operators are available, in which case no plan can be found. It essentially implements a backtracking search algorithm. The first step is to check if there are any operators available (precondition check). If so the first operator encountered is executed and the algorithm recursively calls itself. If no operators are available it proceeds to check for any available methods. If so the subcollection of methods and operators are added to their respective lists and a recursive call is made. If there are also no methods available we go back up one level of recursive calls and try the next available operator or method. Figure 6.1 shows a basic example where a travel from home to park task is decomposed into its methods which are then decomposed into their operators until a path is found where all the preconditions are satisfied. In this case a successful plan is found: call taxi to home, ride taxi to park, pay driver.

6.1.2 The Emotions

Similarly to the choice of planner, the emotions are also represented in a basic form. However, rather then choosing an existing system the decision was made to make our own. Two reasons motivated this decision. The first is due to the complexity and variety of emotion systems it is difficult to chose a neutral baseline that will not significantly bias the results. The second reason is that existing emotion systems often have many levels of abstraction and interaction which makes it difficult to predict and observe how the emotions come into being. By implementing our own system we can make sure to allow for transparent observation and control. As an additional note, for the purposes of the presented model the inputs and outputs of the emotion system are all that is used to integrate the emotions and planning. Therefore, as long as no alterations of the actual emotion system itself are needed to make it compatible, the choice of emotion system should not invalidate the model. The emotions are represented using a class that stores the name, strength, source, and target of each emotion. This is all the basic information that we need to handle emotion generation, interaction, and degradation.

To interact with the emotion system several methods were added. The first group of methods was created to facilitate the management of the emotional state. This includes operations such as generating new emotions, updating emotions according to events and actions, and determining conflicts

FIGURE 6.1: Example of a traveling problem where the 'travel from home to park' task is decomposed into its methods (in blue). The 'travel by taxi' method is then decomposed into its operators (in yellow) [23].

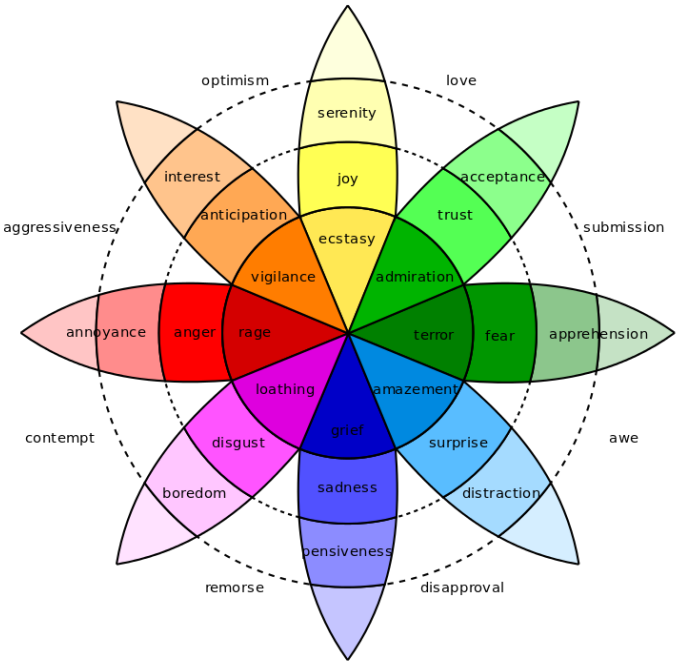


in the emotional state as well as resolving these conflicts. When an operator is executed by the planner these methods can be used to generate an emotion. They will proceed to check whether the agent in question is already experiencing this emotion. If so the strengths are added together, otherwise the strength of the generated emotion is kept. The next step is to check if the generated emotion is in conflict with an existing emotion. If a conflicting emotion is present then the strength of the generated emotion and the conflicting one are compared. The strongest emotion is kept and the weaker one discarded. If there are no conflicting emotions present and the emotion does not exists yet it is simply added to the emotional state.

The conflicts between opposing emotions are determined based on Plutchik's wheel of emotions [26]. From the emotions that can be seen in figure 6.2 we currently support the use of anger, anticipation, joy, trust, fear, surprise, sadness, and disgust. Since social emotions such as shame and guilt are not included, and to maintain the simplicity of the prototype, these currently do not have opposites in the prototype. However, the list of opposites is maintained as a key-value list and can therefore easily be amended as desired.

The second group of methods handles the degradation of emotions over time, including eventually removing the emotions from the emotional state if they have been around for long enough for their strength to completely fade. Time is handled in a discrete fashion and this functionality is mainly there to experiment with how emotion degradation over time can reduce the influence of emotions on the planning as the emotions degrade. For example, an agent can calm down over time. The planner has no knowledge of this idea of time and also does not need to. This method is called every

FIGURE 6.2: Plutchik's wheel of emotions [35].



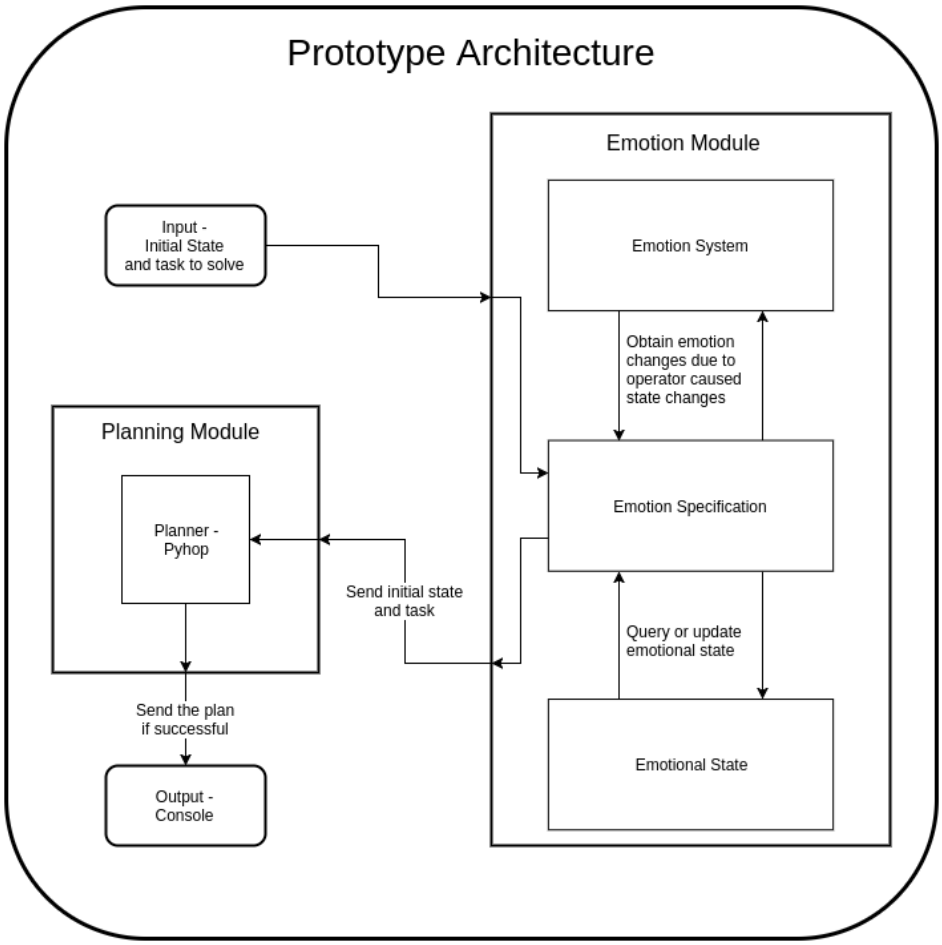
time an action (operator) is executed together with the time that the operator's action takes. The degradation is also based on whether the source or target or both are still present, which are Boolean values stored in each emotion object. When neither the source nor the target are present emotions degrade the fastest and when both are present they degrade the slowest.

6.1.3 The Module Interactions

The planning and emotion module interactions, as specified by the model, are purely focused on passing state related information back and forth. In other words, the planning module sends state changes (due to actions) to the emotion module and the emotion module sends back which actions are available based on the updated state. However, for Pyhop we do not have to explicitly exchange this information. This is because the operators and methods used by Pyhop reside in the emotion specification component. They are after all the actions, which besides their normal actions will also affect the emotional state. Another reason for having the operators and methods reside in the emotion specification component is that they need to have access to the emotional state and emotion system components, and being able to do so without communicating between the planning and emotion module makes the operators and methods far simpler to implement.

In figure 6.3 we can see the prototype architecture. In the case of our prototype we do not need any scaffolding for the planner due to the fact that we can specify actions for the planner as operators and methods, which is done in the emotion specification component. This means the emotion module can simply pass these, together with the initial state and task (goal), to the planner.

FIGURE 6.3: The prototype architecture.



The actual implementation of the scenarios and the emotional reactions is achieved by creating methods and operators in the emotion specification component. The methods contain the decision making logic while the operators represent the actions that actually modify the state of the world and agent. Both are specified with domain specific knowledge, and, in addition to normal decision making logic, the methods also contain emotional reasoning. The emotional reason consists of preconditions that are based on the agent's emotional state. This means that an agent's available methods are limited to the one's whose preconditions are met by the agent's current emotional state. As there are also logic based preconditions, eg. cannot get in a taxi if there is no taxi, cognitive reasoning is no longer the only influence on the decision making process. A method is therefore augmented with preconditions that can contain a combination of cognitive and emotional reasoning. The methods and operators can then be used by the Pyhop planner to attempt to find a plan. The plan that Phyop finds ends up being based on the reasoning contained in the methods rather than trying to reach an explicitly defined goal state. Explicitly stating a goal state is also possible for Phyhop, but our way allows the planning to follows a natural searching process for a better world state. With an explicitly defined goal state the planner will only try to achieve that specific state, ignoring all other properties that are not defined in the goal state.

The intercommunication algorithm, now that the components have been explained, becomes very straightforward. After the methods and operators have been defined as desired all one has to do to try to find a plan is to provide an initial state and one or more tasks that have to be achieved. The planner can then be called with this state and tasks and it will make use of the defined methods and operators to search for a plan. As mentioned, the methods and operators interact with the emotion system and emotional state components to determine additions, updates, and removal of emotions while also performing basic state changes. An example of a basic state change due to an operator is to travel with a taxi to New York, so the location of the agent should be updated to be New York. While emotional reasoning involves sending the new state to the emotion system and updating the emotional state based on its results. For example, the strength of anger towards agent B is increased by 20 points. Anger towards agent B is already present with a strength of 5 so it now becomes a strength of 25.

6.2 The Scenarios

In this section we will provide an overview of each scenario as well as some elaboration on the differences we expect to see in such scenarios when they are run with and without emotions. The section concludes with a discussion of the testing methodology, in which we attempt to determine what constitutes a successful test.

The general goal of the scenarios is to show that even with a prototype that uses a fairly simple planner and emotion system, combining planning and emotions allows for more diverse situations to be handled while also improving the realism of the decisions made by the planner. By tapping into emotions the planner is guided towards decisions that normally would not be considered. It is this added angle of approach that makes the combination interesting.

The scenarios are implemented with three independent variables and one dependent variable. The first independent variable is the collection of methods and operators that are available for the planner. Each scenario has their own collections of methods and operators that does not change. Whether elements of this collection can be used by the planner depends on if their preconditions can be met. The second independent variable is the task (goal) that the planner has to achieve, in our case each scenario has a specific task that has to be solved. The third independent variable is the starting state. This state contains the initial values of any important properties such as the emotional state of our agent. This starting state can be altered to allow each scenario to produce a variety of different plans. Lastly, the dependent variable is the plan produced by Pyhop. Besides checking if a successful plan was generated we can also look at the actions and their ordering to determine the quality of the plan.

6.2.1 Scenario 1

The first scenario is a simple single agent, two person, interaction in which our agent starts by having an object of his taken away by the other person. The starting state can be defined according to, for example, the relationship

between these two. If they are friends the response will most likely be playful or surprised. However, if the other person is a stranger it might spark fear or anger. From here the task given to the planner is to get the object back. Depending on the whether the agent is stronger than the other person, and in combination with the agent's emotional state, a plan is created. This plan can consist of a combination of the actions: ask the object back, demand it back, and take it back.

The main idea behind this scenario is to look into opening up new paths for the planner to explore by including emotional reasoning on top of existing cognitive reasoning. Without emotions, the action selection to get the object back is based on whether the agent is stronger than the other person, and otherwise asking for the object to be given back. With emotions, anger can, for example, allow the agent to choose to take the object back, even when the agent isn't stronger.

6.2.2 Scenario 2

The second scenario simulates a social scenario in which information of a negative nature is revealed about our agent. This brings with it a range of social emotions that the agent has to process while simultaneously trying to relieve the bad situation he finds himself in. The starting state in this case contains several emotions based on the information about the agent that was revealed and how he reacts to the unwilling reveal. This scenario includes emotions such as shame and guilt due to the negative nature of the agent's previous actions. The planner's task is to relieve the negative emotions that are being experienced by the agent. Domain specific knowledge has been added to allow the agent to, for example, reduce the shame he is feeling by explaining his point of view and reasoning for his actions. Using options such as these the planner tries to reach an agreeable emotional state where the negative emotions have dissipated or have low strength values.

Through this scenario we want to demonstrate that by enabling the planner to make decisions based on the emotional state it can resolve a simple social conflict scenario. The idea behind this scenario stems from the exploration of how emotions facilitate and guide our behavior, especially in maintaining healthy social relationships. Incorporating such behavior into a planner should result in a more realistic resolution of the scenario.

6.2.3 Methodology for Testing

In order to determine what constitutes a successful test we have to discuss what the prototype should reveal about the theory discussed in the previous chapters. Part of this criteria has been mention in the previous sections on the reasoning behind each scenario. In the more general sense the prototype has been created to determine if the model specified in chapter 5 can be successfully implemented. Does it enable a planner, in this case Pyhop, to perform emotional reasoning in addition to cognitive reasoning? Which leads to the next question to answer, has the addition of emotional reasoning to an independent planner lead to meaningful improvements to the range of problems it can realistically solve? While realism is difficult to quantify, in this case we are interested in whether the planner will solve problems more akin to how a human would. We will look to see if the

addition of emotions causes the scenarios to be resolved in a manner that takes into account not only the logical but also emotional consequences of actions. In addition, comparing the decisions made by the planner to what we would expect a human to do and experience, as explored in 3, can help us to better identify the realism of the produced plans. The potential for being able to solve a greater variety of problems will be determined by looking at whether Pyhop provides a better solution with emotions when compared to without emotions. A better solution in this case is in terms of whether the addition of emotions creates a more lifelike response to the given situations. However, these tests are not definitive proof that the combination of cognitive and emotional reasoning provided by the model is a generically applicable improvement. They are mainly to show that it is feasible and potentially beneficial to integrate emotions and planning using a generic model.

6.3 Results and Evaluation

In the following section we will first give a short synopsis followed by a step-by-step evaluation of each scenario when run through the prototype. This section will conclude with a results discussion. Here we answer our research questions based on the findings from the prototype testing methodology, while also relating back to the presented model and emotion exploration from the previous chapters.

6.3.1 Scenario 1

In the first scenario the planner is able to successfully create a plan if it is able to convince the other person to give the object back, or, to simply take it back. Without emotions the scenario plays out by having our agent simply consider his options. If he determines he is stronger than the other person he will take the object back, otherwise he will try his other two options of asking the object back followed by demanding it back. In this case the agent only succeeds to get the object back if he is stronger then the other person. This is due the fact that asking the object back always fails and demanding the object back can only work when the agent is angry enough to intimidate the other person successfully.

When we include emotions we see that new possibilities arise while others become unavailable. Anger allows the agent to take the object back even though he is physically weaker, while fear causes the agent to avoid confrontation, guiding the agent towards less aggressive options. Weighing decisions based on the emotions of the agent allows it to make decisions that aren't solely based on whether the agent is strong enough to take it back or not. A large part of human decision making is that we are heavily influenced by our emotional state, often, for better or worse, we are even able to overwrite a logical decision.

In figure 6.4 we have an example of the Pyhop planner trying to find a way to acquire the taken object. The initial state of the agent contains the fear emotion with a strength of 50. The planner starts with the *acquireObject* method, which is the method that contains the bulk of the decision making logic. This method determines that the object is not back in the

FIGURE 6.4: One of the plans output by scenario 1.

```
OPERATORS: demandObjectBack, askForObject, takeBackObject

TASK:         METHODS:
acquireObject acquireObject
    goal1.object = True
** pyhop, verbose=3: **
   state = state1
   tasks = [('acquireObject', 'me')]
depth 0 tasks [('acquireObject', 'me')]
depth 0 method instance ('acquireObject', 'me')
depth 0 new tasks: [('askForObject', 'me'), ('acquireObject', 'me')]
depth 1 tasks [('askForObject', 'me'), ('acquireObject', 'me')]
depth 1 action ('askForObject', 'me')
depth 1 new state:
    state1.emotion = {'me': {'anger': [anger, 45, False, False]}}
    state1.object = {'me': False}
    state1.stronger = True
depth 2 tasks [('acquireObject', 'me')]
depth 2 method instance ('acquireObject', 'me')
depth 2 new tasks: [('demandObjectBack', 'me'), ('acquireObject', 'me')]
depth 3 tasks [('demandObjectBack', 'me'), ('acquireObject', 'me')]
depth 3 action ('demandObjectBack', 'me')
depth 3 new state:
    state1.emotion = {'me': {'anger': [anger, 95, False, False]}}
    state1.object = {'me': False}
    state1.stronger = True
depth 4 tasks [('acquireObject', 'me')]
depth 4 method instance ('acquireObject', 'me')
depth 4 new tasks: [('takeBackObject', 'me'), ('acquireObject', 'me')]
depth 5 tasks [('takeBackObject', 'me'), ('acquireObject', 'me')]
depth 5 action ('takeBackObject', 'me')
depth 5 new state:
    state1.emotion = {'me': {'anger': [anger, 90, False, False]}}
    state1.object = {'me': True}
    state1.stronger = True
depth 6 tasks [('acquireObject', 'me')]
depth 6 method instance ('acquireObject', 'me')
depth 6 new tasks: []
depth 7 tasks []
depth 7 returns plan [('askForObject', 'me'), ('demandObjectBack', 'me'), ('takeBackObject', 'me')]
** result = [('askForObject', 'me'), ('demandObjectBack', 'me'), ('takeBackObject', 'me')]
```

agent's possession and therefore proceed to choose between the potential operators: ask object back, demand object back, and take object back. The agent is fearful and therefore the preconditions for demanding (fear <= 40) and taking (anger strength >= 70) the object back cannot be met. The planner proceeds to choose the only operator whose precondition can be met, the ask object back option. This adds the *askForObject* operator to the Pyhop task list, together with another instance of the *acquireObject* method. The next step in Pyhop is to call the operator that has been added to the task list. The *askForObject* operator returns that the object has not been reacquired and calls the emotion handling method to add anger, with a strength of 50, to the agent's emotional state. The anger emotion is stronger than the currently experienced fear and therefore replaces the fear in the agent's emotional state. Lastly, the operator calls the emotion degradation method, which degrades all emotions in the agent's emotional state according to a time step determined by the operator in question. Pyhop backtracks from the failed operator and the next task in the list is again *acquireObject*. At this point the agent is angry, but not angry enough to outright take the object back. Therefore, the *acquireObject* method resolves in adding the *demandObjectBack* action to the task list, again followed by another *acquireObject* task. Pyhop then executes the *demandObjectBack* operator which in this case fails because the agent isn't angry enough (anger strength >= 60) to intimidate the person into giving the object back. The operator calls emotion handling to add a strong anger emotion with a strength of 80, which in this cases combines with the existing anger emotion to create a very angry agent with anger strength 100. Lastly, the operator calls the emotion degradation method. Pyhop backtracks again and runs the *acquireObject* method again. At this point the agent is so angry that the precondition to take the object back by force has been satisfied. The *acquireObject* method chooses

the highest strength anger response possible, meaning that the *takeBackObject* operator supersedes the *demandObjectBack* operator. The *takeBackObject* operator is added to the task list together with the *acquireObject* method. The *takeObjectBack* operator is executed, and, having successfully intimidated the other person and taken the object back, the *haveObject* Boolean is updated to *True*. One last time the *acquireObject* method is executed, which returns an empty task list since the object has been acquired. Since there are no more tasks to execute Pyhop is done and returns the plan as can be seen at the bottom line of figure 6.4.

6.3.2 Scenario 2

The second scenario's implementation has two methods that enable the planner to handle the scenario. The first method handles the initial reaction to the revelation of the negative information about the agent, while the second method guides the planning with respect to mitigating the negative emotions that resulted from the revelation and, potentially, the initial reaction. The decision to split the emotional reasoning into two parts is due the basic operation of the Pyhop planner. It simply attempts to find a solution and does not attempt to optimize in any way. Therefore, with the reasoning that this is domain specific knowledge, we simulate that the initial reaction should be rushed and less thought out. Then, after having a moment to consider the situation, the other method takes over and guides the mitigation process in a more structured manner. This also allows us to create operators that perform general and specific reactions depending on what the scenario requires.

In this scenario the planner is regularly taken over by emotional reasoning, this is due to the situation having resulted in an overflow of powerful feelings. After the initial reaction things calm down and the planner, still dominated by the emotions, chooses actions that can relieve the negative emotions. Logic has been given a back seat in this scenario. Instead emotions are the preconditions of how the agent will react to the accusations. The scenario was inspired by the exploration of the function of social emotions in humans. It attempts to simulate the guidance behavior that stems from strong emotions, and the potential of these emotions to resolve social conflict.

In figure 6.5 the planner has to find a plan that will mitigate the negative emotions, caused by the initial situation, as best as possible. Unlike the first scenario, the method and operator preconditions in the second scenario are based solely on emotions. This was done to experiment with a plan being created based on social emotions, and whether this could resolve a social conflict in an acceptable manner. The planner can choose operators that perform actions that will reduce the strength of certain emotions. By combining these operators the planner's task is to find an emotional state that does not contains any high strength negative emotions.

The Pyhop task list starts off containing the *initialReaction* and *relieveNegativeEmotions* methods. The starting emotional state contains anger (strength 50), shame (strength 80), and guilt (strength 60), each with high values of 50+. To start, Pyhop executes the *initialReaction* method, which contains emotional reasoning in which rash emotional decisions are made. If anger and disgust are strong this method will cause the *leave* operator to be added,

FIGURE 6.5: One of the plans output by scenario 2.

```
OPERATORS: demandObjectBack, askForObject, takeBackObject

TASK:         METHODS:
acquireObject acquireObject
    goal1.object = True
** pyhop, verbose=3: **
   state = state1
   tasks = [('acquireObject', 'me')]
depth 0 tasks [('acquireObject', 'me')]
depth 0 method instance ('acquireObject', 'me')
depth 0 new tasks: [('askForObject', 'me'), ('acquireObject', 'me')]
depth 1 tasks [('askForObject', 'me'), ('acquireObject', 'me')]
depth 1 action ('askForObject', 'me')
depth 1 new state:
    state1.emotion = {'me': {'anger': [anger, 45, False, False]}}
    state1.object = {'me': False}
    state1.stronger = True
depth 2 tasks [('acquireObject', 'me')]
depth 2 method instance ('acquireObject', 'me')
depth 2 new tasks: [('demandObjectBack', 'me'), ('acquireObject', 'me')]
depth 3 tasks [('demandObjectBack', 'me'), ('acquireObject', 'me')]
depth 3 action ('demandObjectBack', 'me')
depth 3 new state:
    state1.emotion = {'me': {'anger': [anger, 95, False, False]}}
    state1.object = {'me': False}
    state1.stronger = True
depth 4 tasks [('acquireObject', 'me')]
depth 4 method instance ('acquireObject', 'me')
depth 4 new tasks: [('takeBackObject', 'me'), ('acquireObject', 'me')]
depth 5 tasks [('takeBackObject', 'me'), ('acquireObject', 'me')]
depth 5 action ('takeBackObject', 'me')
depth 5 new state:
    state1.emotion = {'me': {'anger': [anger, 90, False, False]}}
    state1.object = {'me': True}
    state1.stronger = True
depth 6 tasks [('acquireObject', 'me')]
depth 6 method instance ('acquireObject', 'me')
depth 6 new tasks: []
depth 7 tasks []
depth 7 returns plan [('askForObject', 'me'), ('demandObjectBack', 'me'), ('takeBackObject', 'me')]
** result = [('askForObject', 'me'), ('demandObjectBack', 'me'), ('takeBackObject', 'me')]
```

resulting in a situation where the plan is simply to leave, ending the scenario.

In the example's starting state, however, the agent makes a less extreme decision, due to not being outraged, and the *initialReaction* method adds the *collectSelf* operator to the task list to mediate the high strength anger that is currently a being experienced by the agent. The *collectSelf* operator is then executed by Pyhop, which calls the emotion handling method with a negative strength value to be added to the anger emotion. This is to simulate the agent calming itself down. Combined with the call to the emotion degradation method at the end of the *collectSelf* operator anger's strength is reduced to zero and removed from the emotional state. Shame (strength 79) and guilt (strength 59) are still left in the agent's emotional state at this point. The next task is the *relieveNegativeEmotions* method. This method looks at the agent's emotional state and chooses operators that will attempt to reduce a specific emotion if its strength value exceeds 50. In this case the next strongest emotion felt is shame, and therefore the *explainSituation* operator is added to the task list, together with the *relieveNegativeEmotions* method. The *explainSituation* operator is then executed, which contains a call to the emotion handling method to reduce the shame felt by the agent, followed by a call to the emotion degradation method the shame emotion is now at strength 28. Next up is the *relieveNegativeEmotions* method again. At this point shame has been reduced to below the threshold to trigger the *explainSituation* operator and instead the *makeAmends* operator is added to relieve the still strong guilt (strength 58) emotion being experienced by the agent. Again the *relieveNegativeEmotions* methods is also added. Pyhop proceeds to execute the *makeAmends* operator, which in turn reduces the strength of the guilt emotion to eight. This operator also ends with a call to the emotion degradation method. One last time the *relieveNegativeEmotions*

method is executed to find that none of the emotions exceed the threshold that has been set, resulting in an empty Pyhop task list, which then results in Pyhop returning the plan as can be seen at the bottom of figure 6.5.

6.3.3 Results

First a reminder of the research questions. Research question 1 asks if the integration of emotional reasoning can allow existing automated planners to realistically solve social scenarios. We then follow up this question with research question 2 which concerns itself with to what extent the exploration of emotion models and planners can lead to generic integration of emotion models and planners. Next, we will discuss the results of the prototype scenarios as well as some conclusions that we can draw from them using the testing methodology.

We have implemented the prototype according to how our model describes the interactions between a planning and emotion module. We have contained the planning and emotions to their respective modules while allowing the emotions to influence the planner by filtering actions based on the emotional state of the agent. In the first scenario we can see that the emotional state of the agent changes the planning behavior of the planner by enabling and disabling certain actions based on whether their emotion based preconditions are met. This action filtered results in more focused behavior which, if the emotion model is correct, will cause the decision making to be more realistic. Without emotions the planner only has the choice to take the object back, regardless of the social consequences, which it cannot take into account without emotions. However, when the proper emotions are included, the agent can evaluate the situation based on their personal desire to reacquire the object. If the agent truly desires the object back he will become angry when it is taken away, which allows for more decisive action to be undertaken. However, if the object is not of much value the emotional state will reflect this, causing the agent to take a calmer approach.

In the second scenario this is taken a step further. Here actions are chosen purely based on the emotional state of the agent in an attempt to allow the social emotions to successfully resolve a social conflict. As mentioned this scenario is motivated by the fact that social emotions play an important role in human (social) behavior. These emotions allow us to take into account the beliefs, desires, and intentions of other individuals. By recognizing these aspects of one's self and others, better decisions can be made. In this scenario we can see that allowing an agent to plan according to it's emotional state allows it to engage in actions that relieve the negative social emotions. This is very similar to how humans interact with each other. The social emotions of shame and guilt are caused by the emotion model being aware of others and our agent's relationship with them. By reacting to it's own emotional state the agent is then able to choose the proper course of action, which not only relieves his own emotional distress, but hopefully, also resolves the negative view that the others have of him.

The first research question is composed of two parts that we want to answer. Firstly, can emotional reasoning be integrated with existing automated planners, and secondly, does this allow the planner to realistically solve social scenarios. The prototype implementation shows, through the

addition of emotion-based filtering of actions, that such an integration is indeed possible. Though to what extent this transfers to other planners and the success that can be achieved is part of the second research question. As we have described in the testing methodology, we see realism as generating plans that seem logical to a human observer, and we do believe these results to be more pleasing. In addition we also wish to see if the emotions allow for a better solution when compared to without emotions. While the first scenario shows a limited example of the same situation when approached with and without emotions it does show that decisions are more akin to the process of a human. The emotional state in this case reflects the importance of the lost object, which in turn influences the amount of effort taken to retrieve it. The scenario demonstrates this by having the agent only attempt actions that corresponds to it's level of emotional involvement, in this case mainly anger due to the loss of the object. Without emotions however, it becomes a simple formula of whether or not the agent is strong enough to take it back. There are simply no other logical factors to consider. This kind of consideration of actions and consequences is one of the main reasons for attempting to introduce emotional reasoning to planners. The second scenario reinforces this ability of emotions to guide behavior by allowing emotions to take control of the decision making process. When the agent is presented with negative emotions it attempts to relieve these feelings in any way it knows how. As these emotions can be generated by an emotional model that properly identifies the social situation, the planner can receive the ability to successfully resolve a social conflict by simply following its emotional reasoning. Of course this does not mean that the choices made are the most optimal. However, it does seem appropriate for the agent to simply storm out and leave the scene if it feels it is being wrongfully accused of something. Via these two scenarios we have shed some light on how cognitive and emotional reasoning can be coerced to result in more pleasing plans of action.

The model that was presented in 5, in combination with the emotion discussion in 4 are most the important components when attempting to answer the second research question. The model is based on emotion research that explores not only what the effects of emotions on planning are, but also the methods via which emotions influence human planning. By drawing inspiration from humans and emotion research, a model was created that attempts to mimic the human combination of cognitive and emotional reasoning. In order to make the model as generic as possible we identified three points of influence that most planner have access to: actions, task decomposition, and goals. It is hard to imagine a planner without actions or goals, however, task decomposition is less generic. This is alleviated by the fact that the model can implement most of the emotions without this component. Although the model does benefit from having more refined control through task decomposition. The prototype, for example, is able to function without task decomposition by simply filtering the actions based on preconditions that require certain emotions to be present in the emotional state. However, there are planners that can make the actual implementation of an integration model as proposed rather difficult. Planners that, for example, already keep track of other agents' internal state can be difficult to integrate with an emotion model that bases its emotions on such internal states. While in theory these can be kept separate, in practice keeping two

copies of such information, especially when acquired in different manners could result in unexpected and undesired behavioral repercussions. Another issue could arise in the other direction. If the planner is self contained it can be difficult to extract the state between planning steps. The state that a proper emotion model will most definitely require in order to generate correct emotions. To answer the second research question on generic applicability of integration methods. The presented model has two hard conditions that should be met. Firstly, the state, including between planning steps, has to be exchangeable between the planner and emotion model. Secondly, the planner action selection has to be influenceable, either by allowing the set of possible actions to be filtered or by modifying actions directly.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis we have explored the function that emotions serve for humans, their effects on our decision making process, and how this can be translated to influencing the decisions an automated planner makes. We found that there was a vast amount of knowledge to be found on how emotions and planning work. However, there is little integration to be found in terms of a planning system that incorporates emotions to achieve a system that performs cognitive and emotional reasoning simultaneously.

To explore the potential of introducing emotions to planners we first performed research to determine what kind of systems already exist. While there was the occasional system that dabbled in this integration, they are build from the ground up to achieve this. From our exploration of emotions and how they influence human decision making, we became motivated to explore the possibility of integrating existing emotion systems and planners. The main benefit would be the ability to take advantage of the work that has been done in both fields. The exploratory effort led to a model that defined three planning components, which most planners have, that can be influenced by emotions. Next, we specified how to potentially implement the model, with special care to keep the emotion and planning system details generically applicable. Lastly, we created a prototype that implements the model using a basic, self-made, emotion system in combination with the Pyhop HTN planner.

Through the exploration of emotions, human decision making, and automated planners we defined a model. By then implementing and testing a prototype using several scenarios we attempted to demonstrate the viability of generic integration of emotions and planning. By analyzing the model according to our testing methodology, and running several scenarios on the prototype, we were able to determine that for basic emotion and planning systems generic integration is possible and beneficial. In terms of beneficial, our results showed that a basic scenario with simple emotions such as anger and fear can already enhance the planning significantly. We believe this to be due to, as highlighted by other research, the importance of emotions in the human decision making process. In addition our results revealed that even in the case of a simple social conflict scenario the planner is able to plan according to emotional impulses to bring the situation to a more natural conclusion. Again this stems from research that discovered that social emotions often play a guidance role, allowing our basic planner to plan with the knowledge endowed by the emotion system. By experimenting, developing and testing a model that attempts to generically

integrate basic planning and emotion systems we have shown that not only is such integration feasible, but it can significantly enhance the realism of resulting plans by allowing an existing planner to take into account a whole new spectrum of information.

7.2 Future Work

There is plenty of work left to be done in the area of combining planning and emotions. In this paper we have only scratched the surface of the possibilities. Delving into a more complex planner, such as an epistemic planner that works with beliefs, and finding a way to integrate emotions into such as planner could raise it to new levels of human planning simulation and would be the next logical step. Making use of an established emotion framework to accurately determine what emotions should be generated by the events that are taking place around the agent could also lead to more realistic reactions by automated planners. Eventually, finding an integration method that works with domain-independent planners could allows the planning and emotion research to be combined into truly realistic automated planners.

Bibliography

[1] Christina Bartl and Dietrich Dörner. "PSI: A theory of the integration of cognition, emotion and motivation". In: *Proceedings of the 2nd European Conference on Cognitive Modelling*. DTIC Document. 1998, pp. 66–73.

[2] Monica Y Bartlett and David DeSteno. "Gratitude and prosocial behavior helping when it costs you". In: *Psychological science* 17.4 (2006), pp. 319–325.

[3] Roy F Baumeister et al. "How emotion shapes behavior: Feedback, anticipation, and reflection, rather than direct causation". In: *Personality and Social Psychology Review* 11.2 (2007), pp. 167–203.

[4] Gerald L Clore and Andrew Ortony. "The semantics of the affective lexicon". In: *Cognitive perspectives on emotion and motivation*. Springer, 1988, pp. 367–397.

[5] Antonio R. Damasio. *Looking for Spinoza: joy, sorrow, and the feeling brain*. Harcourt, 2003.

[6] Jean Decety and Kalina J Michalska. "Neurodevelopmental changes in the circuits underlying empathy and sympathy from childhood to adulthood". In: *Developmental science* 13.6 (2010), pp. 886–899.

[7] Joao Dias, Samuel Mascarenhas, and Ana Paiva. "Fatima modular: Towards an agent architecture with a generic appraisal framework". In: *Emotion Modeling*. Springer, 2014, pp. 44–56.

[8] Dietrich Dörner and C Dominik Güss. "PSI: A computational architecture of cognition, motivation, and emotion." In: *Review of General Psychology* 17.3 (2013), p. 297.

[9] Paul Ekman. "The argument and evidence about universals in facial expres-sions". In: *Handbook of social psychophysiology* (1989), pp. 143–164.

[10] Agneta H Fischer and Ira J Roseman. "Beat them or ban them: the characteristics and social functions of anger and contempt." In: *Journal of personality and social psychology* 93.1 (2007), p. 103.

[11] Nico H Frijda, Peter Kuipers, and Elisabeth Ter Schure. "Relations among emotion, appraisal, and emotional action readiness." In: *Journal of personality and social psychology* 57.2 (1989), p. 212.

[12] Jonathan Gratch. "Why you should buy an emotional planner". In: *Proceedings of the Autonomous Agents 1999 Workshop on Emotion-based Agent Architectures (EBAA'99)*. Citeseer. 1999.

[13] Joseph Henrich and Francisco J Gil-White. "The evolution of prestige: Freely conferred deference as a mechanism for enhancing the benefits of cultural transmission". In: *Evolution and human behavior* 22.3 (2001), pp. 165–196.

[14] Charles Hoch. "Emotions and planning". In: *Planning Theory & Practice* 7.4 (2006), pp. 367–382.

[15] Yi-Cheng Huang, Bart Selman, Henry Kautz, et al. "Control knowledge in planning: benefits and tradeoffs". In: 1999.

[16] Mark D Johnston and Glenn Miller. "Spike: Intelligent scheduling of hubble space telescope observations". In: *Intelligent Scheduling* (1994), pp. 391–422.

[17] Jennifer S Lerner et al. "Emotion and decision making". In: *Psychology* 66 (2015).

[18] Michael Lewis et al. "Cultural differences in emotional responses to success and failure". In: *International Journal of Behavioral Development* (2009).

[19] Mei Yii Lim et al. "Creating adaptive affective autonomous NPCs". In: *Autonomous Agents and Multi-Agent Systems* 24.2 (2012), pp. 287–311.

[20] Sonja Lyubomirsky, Laura King, and Ed Diener. "The benefits of frequent positive affect: does happiness lead to success?" In: *Psychological bulletin* 131.6 (2005), p. 803.

[21] Michael E McCullough, Jo-Ann Tsang, and Robert A Emmons. "Gratitude in intermediate affective terrain: links of grateful moods to individual differences and daily emotional experience." In: *Journal of personality and social psychology* 86.2 (2004), p. 295.

[22] Claude Harold Miller. "Indignation, defensive attribution, and implicit theories of moral character". In: (2000).

[23] Dana Nau. "Game applications of HTN planning with state variables". In: *Planning in Games: Papers from the ICAPS Workshop*. 2013.

[24] Dana S Nau. "Current trends in automated planning". In: *AI magazine* 28.4 (2007), p. 43.

[25] W Gerrod Parrott and Richard H Smith. "Distinguishing the experiences of envy and jealousy." In: *Journal of personality and social psychology* 64.6 (1993), p. 906.

[26] Robert Plutchik. "The Nature of Emotions Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice". In: *American Scientist* 89.4 (2001), pp. 344–350.

[27] Anand S Rao and Michael P Georgeff. "Modeling rational agents within a BDI-architecture." In: *KR* 91 (1991), pp. 473–484.

[28] Paul Rozin, Jonathan Haidt, and Clark R McCauley. "Disgust." In: (2008).

[29] Gerd Ruebenstrunk. *Structure of OCC Model Emotion Types*. [Online; accessed December 28, 2016]. 1998. URL: `http://www.ruebenstrunk.de/emeocomp/36fa2f7a.jpg`.

[30] Stuart Jonathan Russell et al. *Artificial intelligence: a modern approach*. Vol. 2. Prentice hall Upper Saddle River, 2003.

[31] Deborah A Small, George Loewenstein, and Paul Slovic. "Sympathy and callousness: The impact of deliberative thought on donations to identifiable and statistical victims". In: *Organizational Behavior and Human Decision Processes* 102.2 (2007), pp. 143–153.

[32] Craig A Smith and Richard S Lazarus. "Emotion and adaptation." In: (1990).

[33] June Price Tangney et al. "Are shame, guilt, and embarrassment distinct emotions?" In: *Journal of personality and social psychology* 70.6 (1996), p. 1256.

[34] TL Van der Weide. "Arguing to motivate decisions". In: (2011).

[35] Wikipedia, the free encyclopedia. *Plutchik's Wheel of Emotions*. [Online; accessed December 22, 2016]. 2011. URL: `https://upload.wikimedia.org/wikipedia/commons/thumb/c/ce/Plutchik-wheel.svg/757px-Plutchik-wheel.svg.png`.

[36] Monte Zweben, Michael Deale, and Robert Gargan. "Anytime rescheduling". In: *Proceedings of a Workshop on Innovative Approaches to Planning, Scheduling and Control*. 1990, pp. 251–259.